



**ANALYSIS OF ROUTING WORM INFECTION RATES ON AN IPV4  
NETWORK**

THESIS

James E. Gorsuch, First Lieutenant, USAF

AFIT/GCS/ENG/07-04

**DEPARTMENT OF THE AIR FORCE  
AIR UNIVERSITY**

**AIR FORCE INSTITUTE OF TECHNOLOGY**

---

---

**Wright-Patterson Air Force Base, Ohio**

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED



The views expressed in this thesis are those of the author and do not reflect the official policy or position of the United States Air Force, Department of Defense, or the U.S. Government.

AFIT/GCS/ENG/07-04

**ANALYSIS OF ROUTING WORM INFECTION RATES ON AN IPV4  
NETWORK**

THESIS

Presented to the Faculty

Department of Electrical and Computer Engineering

Graduate School of Engineering and Management

Air Force Institute of Technology

Air University

Air Education and Training Command

In Partial Fulfillment of the Requirements for the

Degree of Master of Science in Computer Science

James E. Gorsuch, BS

First Lieutenant, USAF

March 2007


APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

**ANALYSIS OF ROUTING WORM INFECTION RATES ON AN IPV4  
NETWORK**

James E. Gorsuch, BS

First Lieutenant, USAF

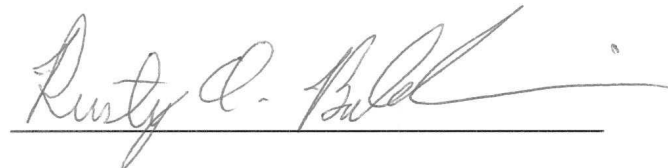
Approved:



Barry E. Mullins, Ph.D. (Chairman)

9 Mar 07

Date



Rusty O. Baldwin, Ph.D. (Member)

9 Mar 07

Date



Richard A. Raines, Ph.D. (Member)

9 Mar 07

Date

## Abstract

Malicious logic, specifically worms cost network users an enormous amount of time and money. Worms, like Slammer and Code Red, infect thousands of systems and denied whole networks access to the Internet. This research examines the ability of the original Slammer worm, a Slammer based routing worm, and a new Single Slash Eight (SSE) routing worm to infect vulnerable systems within a given address space. The ability of Slammer to generate a uniform random IP addresses in a given address space is established. Finally, a comparison of the speed increase from a worm on a computing system in 2003 to those available today is performed.

Both the Slammer based routing worm and the SSE routing worm spread faster than the original Slammer. The random number generator of the original Slammer worm generates a statistically uniform distribution of addresses within the range under test. Furthermore, despite the previous research into the speed of worm propagation, there is still a need to test worms on the current systems. The speed of the computing systems that the worms operated on in the past were more than three times slower than today's systems. As the speed of computer systems continue to grow, the speed of worm propagation should increase with it as their scan rates directly relate to their infection rate. As such, any inherent immunity of an IPv6 network from scanning worms should be reexamined.

*Thanks to  
God  
For all He has given me,  
and  
For my wife  
One of God's many gifts,  
For without her love and understanding  
My life would be naught.*

## Acknowledgments

I would like to express my sincere appreciation to my faculty advisor, Dr. Barry E. Mullins, my committee members Dr. Rusty O. Baldwin and Dr. Richard A. Raines for their guidance and support throughout the course of this thesis effort. The help of Major David M. Kaziska during the final weeks of my thesis writing was a Godsend and I owe him a large debt of gratitude. All of the Instructors at the Air Force Institute of Technology (AFIT) were top-notch and helped in too many ways to be covered here.

Additionally, I need to heap thanks on all of fellow students who helped me survive my time at the AFIT. I have truly stood on the shoulders of giants. Without their help I would have drowned under the work and been left behind. There are so many students that have become my friends and played significant roles in my survival of AFIT. They all deserve to be named, but naming them all and praising their aid would be another thesis. For those not named individually, you know who you are, and I will always remember the constant help and support you gave me. Specifically, I need to thank Lt Charles Fetzek for assisting in the last minute editing and proofreading of this piece of work. The leadership, guidance, and friendship of our section leader Maj. Timothy Franz was a stable and welcome influence during my tour at AFIT. Capt. Sean Hudson's friendship and aid during our many shared group projects will not be forgotten.

Finally, I would be remiss if I failed to mention the complete appreciation I have for the sacrifices my lovely wife and son had to endure due to this life experience at AFIT. The long hours of studying, loss of focus on family, and the final weeks of almost complete isolation required to finish this thesis was handled without serious complaint and with the loving support I have come to know from them both. Thank you all!

James E. Gorsuch



## Table of Contents

Abstract .....	v
Table of Contents .....	viii
List of Figures .....	xii
List of Tables .....	xv
I. Introduction.....	1
1.1 Motivation .....	1
1.2 Overview and Goals .....	3
1.3 Thesis Overview .....	3
1.4 Summary.....	5
II. Literature Review .....	6
2.1 Introduction .....	6
2.2 Worms .....	6
2.2.1 Pseudo Random Number Generation in Worms.....	9
2.2.2 Code Red.....	10
2.2.3 Slammer .....	11
2.2.4 The Border Gateway Protocol (BGP) and /8 Routing Worms .....	12
2.2.5 The BGP Routing Worm Propagation .....	14
2.3 Internet Modeling .....	17
2.4 Internet Protocol Addressing.....	19
2.5 Increasing Speed of Technology .....	21
2.6 Related Research .....	21
2.7 Summary.....	26

III. Methodology .....	27
3.1 Introduction .....	27
3.2 Problem Definition .....	27
3.2.1 Goals and Hypothesis .....	27
3.2.2 Approach.....	28
3.2.3 Assumptions and Limitations .....	29
3.3 System Boundaries .....	29
3.4 System Services.....	30
3.5 Workload .....	31
3.6 Performance Metrics .....	31
3.7 Parameters .....	31
3.7.1 System Parameters .....	31
3.7.2 Workload Parameters.....	33
3.8 Factors .....	33
3.9 Evaluation Technique.....	34
3.10 Experimental Design .....	34
3.10.1 Network Configuration and System Infection Procedure.....	34
3.10.2 Experimental Exploration of Slammer Randomness.....	35
3.10.3 Matlab Model of Infection Rate.....	38
3.10.4 Experimental Validation of Matlab Model Infection Rate Simulation.....	40
3.10.5 Experimental Examination of the Slammer Scanning Rate.....	46
3.10.6 Examination of the Slammer Infection Doubling Rate.....	47

3.11 Analysis and Interpretation of Results .....	48
3.12 Summary.....	50
IV. Analysis and Results.....	51
4.1 Slammer Packet Generation .....	51
4.2 Slammer Randomness .....	54
4.2.1 Slammer IP Address Randomness .....	54
4.2.2 Slammer Octet Randomness .....	58
4.3 Matlab Model Simulation of Slammer Routing Worm Operation.....	68
4.3.1 Validation of Matlab Model Infection Rate Simulation .....	68
4.3.2 Matlab Model of Doubling Rate versus Observed Slammer Rate.....	75
4.3.3 Matlab Model versus Wei Slammer Infection Rate.....	81
4.4 Scanning Worms in a Computing Architecture of Today .....	83
4.5 Single Slash Eight (SSE) Routing Worm.....	89
4.5.1 SSE Routing Worm Creation.....	89
4.5.2 Matlab Model of the SSE Routing Worm.....	90
4.5.3 SSE Routing Worm Infection Rate Comparison .....	90
4.6 Summary.....	93
V. Conclusions and Recommendations .....	95
5.1 Restatement of the Problem and Conclusions.....	95
5.2 Contributions and Significance of Research .....	96
5.3 Recommendations for Future Research.....	97
5.4 Summary.....	98
Appendix A.....	99

A.1 Experiment Hardware .....	99
A.2 Experiment Software .....	100
Appendix B .....	101
B.1 Slammer Code.....	<b>Error! Bookmark not defined.</b>
B.1.1 Slammer Code with ASCII .....	<b>Error! Bookmark not defined.</b>
B.1.2 Disassembly of Slammer Code .....	<b>Error! Bookmark not defined.</b>
B.2 Slammer Stack Manipulation.....	<b>Error! Bookmark not defined.</b>
B.3 Slammer Code Corrections.....	<b>Error! Bookmark not defined.</b>
Appendix C .....	102
C.1 Matlab Model Infection Rate Simulation Code.....	102
Appendix D .....	105
D.1 Slammer UDP Packet Example.....	<b>Error! Bookmark not defined.</b>
D.2 Slammer UDP Packet Header Breakout.....	<b>Error! Bookmark not defined.</b>
D.3 Slammer UDP Packet Multicast Detail .....	<b>Error! Bookmark not defined.</b>
Appendix E .....	106
E.1 SSE Routing Worm Assembly Code .....	<b>Error! Bookmark not defined.</b>
Bibliography.....	107
Vita.....	111

## List of Figures

Figure	Page
1. Slammer Global Infection .....	2
2. Zou Code Red Worm Simulation .....	15
3. Zou Slammer Worm Simulation .....	17
4. Pattern of Daily Network Traffic .....	19
5. Wei Slammer Worm Simulation .....	24
6. Wei Slammer Worm propagation with Network Congestion .....	25
7. Network Under Attack .....	30
8. Network Configuration .....	35
9. Netcat Infection Command .....	35
10. Statistical Model Graph Example .....	37
11. Matlab Model Infection Rate Example .....	40
12. Lag Plot of Slammer-Generated IP Addresses .....	55
13. Residual Plots of Slammer IP Address Generation .....	57
14. Fitted Line Plot of Slammer versus Statistical Model .....	58
15. Lag Plot of Slammer-Generated Second Octet .....	59
16. Residual Plot for Slammer-Generated Second Octet .....	61
17. Fitted Line Plot for Slammer-Generated Second Octet .....	62
18. Lag Plot of Slammer-Generated Third Octet .....	63
19. Lag Plot of Slammer-Generated Fourth Octet .....	64
20. Residual Plot for Slammer-Generated Third Octet .....	66

Figure	Page
21. Residual Plot for Slammer-Generated Fourth Octet.....	66
22. Fitted Line Plot for Slammer-Generated Third Octet .....	67
23. Fitted Line Plot for Slammer-Generated Fourth Octet .....	67
24. Zou Code Red versus Matlab Model Code Red Infection Rates .....	69
25. Zou Slammer Worm versus Zou Slammer Routing Worm .....	70
26. Matlab Model of Zou Slammer Routing Worm @ 3,108 pps .....	71
27. Matlab Model of Original Slammer Worm @ 4,000 pps .....	72
28. Matlab Model of Zou Slammer Routing Worm @ 4,000 pps .....	72
29. Matlab Model Composite of Slammer Infection Rates .....	73
30. Matlab Model Slammer Worms versus Zou Slammer Worms.....	74
31. Matlab Model-Generated Doubling Rate.....	76
32. Slammer Doubling Rate.....	77
33. Matlab Model versus Observed Slammer Doubling Rate .....	78
34. Matlab Model versus Slammer Doubling Rate Detailed .....	79
35. Matlab Model-Generated Slammer Worm 2003 .....	80
36. Wei Slammer Worm Simulation.....	82
37. Matlab Model Slammer Worm versus Wei Slammer Worm.....	83
38. Zou Slammer Worm versus Matlab Model Slammer Worm 2003 .....	84
39. Matlab Model-Generated Slammer Worm 2003 .....	86
40. Matlab Model-Generated Slammer Worm 2007 .....	87
41. Matlab Model-Generated Slammer Routing Worm 2003 .....	87

Figure	Page
42. Matlab Model-Generated Slammer Routing Worm 2007 .....	88
43. Matlab Model-Generated Slammer Worms and Slammer Routing Worms.....	88
44. SSE Routing Worm Assembly Code .....	<b>Error! Bookmark not defined.</b>
45. Matlab Model-Generated SSE Routing Worm 2003 .....	91
46. Matlab Model-Generated SSE Routing Worm 2007 .....	92
47. Matlab Model SSE Routing Worms versus Slammer Worms.....	92
48. Slammer Code with ASCII .....	<b>Error! Bookmark not defined.</b>
49. Slammer UDP Packet Example .....	<b>Error! Bookmark not defined.</b>
50. Slammer Infection Packet Samples .....	<b>Error! Bookmark not defined.</b>

## List of Tables

Table	Page
1. Zou Worm Simulation Variables .....	17
2. Netcat Infection Command Description .....	35
3. Statistical Model Example .....	37
4. Experiment Statistical Model.....	38
5. Matlab Model Infection Rate Example.....	39
6. Matlab Model Variables for Code Red Worms .....	42
7. Matlab Model Variables for Zou Slammer Worms .....	43
8. Matlab Model Variables for the Slammer Worms.....	44
9. Matlab Model Variables for the Slammer Routing Worms.....	45
10. Matlab Model Variables for the SSE Routing Worms .....	46
11. Worm Variables for Year Simulations .....	47
12. Slammer Packet Per Second on A/C Power .....	52
13. Slammer Packet Per Second on Battery Power .....	52
14. Slammer IP Address Generation Regression Analysis.....	56
15. Slammer-Generated Second Octet Regression Analysis .....	60
16. Slammer-Generated Third Octet Regression Analysis .....	64
17. Slammer-Generated Fourth Octet Regression Analysis .....	64
18. Matlab Model Variables for 2003 versus 2007 Worm Comparison.....	86
19. Matlab Model of 2003 versus 2007 Infection Rates.....	89
20. Matlab Model Variables for SSE Routing Worm.....	90



Table	Page
21. SSE Routing Worm Speed Comparison .....	93
22. Experiment Computer Specifications .....	99
23. Port Switch Specifications .....	99
24. Experiment Software Versions .....	100
25. Opening Portion of Slammer Code.....	<b>Error! Bookmark not defined.</b>
26. Payload Fix-up of Slammer Code.....	<b>Error! Bookmark not defined.</b>
27. IAT and Process Locating of Slammer Code .....	<b>Error! Bookmark not defined.</b>
28. Send To Protocol of Slammer Code .....	<b>Error! Bookmark not defined.</b>
29. IP Address Generation and Loop of Slammer Code....	<b>Error! Bookmark not defined.</b>
30. Stack Contents Through Command B2 .....	<b>Error! Bookmark not defined.</b>
31. Stack Contents Through Command E2.....	<b>Error! Bookmark not defined.</b>
32. Stack Contents Through Command ED.....	<b>Error! Bookmark not defined.</b>
33. Stack Contents Through Command 107 .....	<b>Error! Bookmark not defined.</b>
34. Stack Contents Through Command 123 .....	<b>Error! Bookmark not defined.</b>
35. Stack Contents Through Command 12B .....	<b>Error! Bookmark not defined.</b>
36. Stack Contents Through Command 136.....	<b>Error! Bookmark not defined.</b>
37. Stack Contents Through Command 174.....	<b>Error! Bookmark not defined.</b>
38. Slammer Random Number Generator Code Corrections .....	<b>Error! Bookmark not defined.</b>
39. Matlab Model Experiment Variable Values .....	103
40. Slammer UDP Packet Header Description .....	<b>Error! Bookmark not defined.</b>

41. Slammer UDP Packet Destination IP Address Detail..**Error! Bookmark not defined.**

Table

Page

42. SSE Routing Worm Code .....**Error! Bookmark not defined.**

# **ANALYSIS OF ROUTING WORM INFECTION RATES ON AN IPV4 NETWORK**

## **I. Introduction**

### **1.1 Motivation**

The dream of one world, one community has all but become a reality as a network of computers now connects the world both virtually and physically via the Internet. Though originally implemented as a way to share information between universities, the Internet has grown to encompass every nation. This has allowed for an amazing sharing of information and resources across the globe. However, with this great good also comes the bad. The global community's interconnectedness and reliance on the Internet has led to many using the Internet for nefarious purposes. There are people using the Internet to perform corporate espionage, steal identities, and in general, create havoc. To that end, one of the most costly sources of this havoc for businesses and users alike on the Internet is malicious logic. Of all the forms of malicious logic, computer worms have shown themselves to be one of the most costly for the Internet community.

Worms can infect thousands of systems in just minutes. These fast infection rates reduce or eliminate the access of large corporations to the average person to Internet services. Slammer, also known as Sapphire and SQL Slammer, was one of the fastest worms ever released onto the Internet. As shown in Figure 1, Slammer spread throughout the world in just minutes. This figure identifies the areas of the world infected by Slammer in less than 30 minutes [MPW03]. The blue infection circles do not accurately represent the number of systems infected per area, but identifies the areas

covered in order to limit the overlap with adjacent zones [MPW03]. This coverage is consistent with every major technological center and city across the globe and illustrates how interconnected the world was in 2003. Since then, the global community has grown larger and even more interconnected.

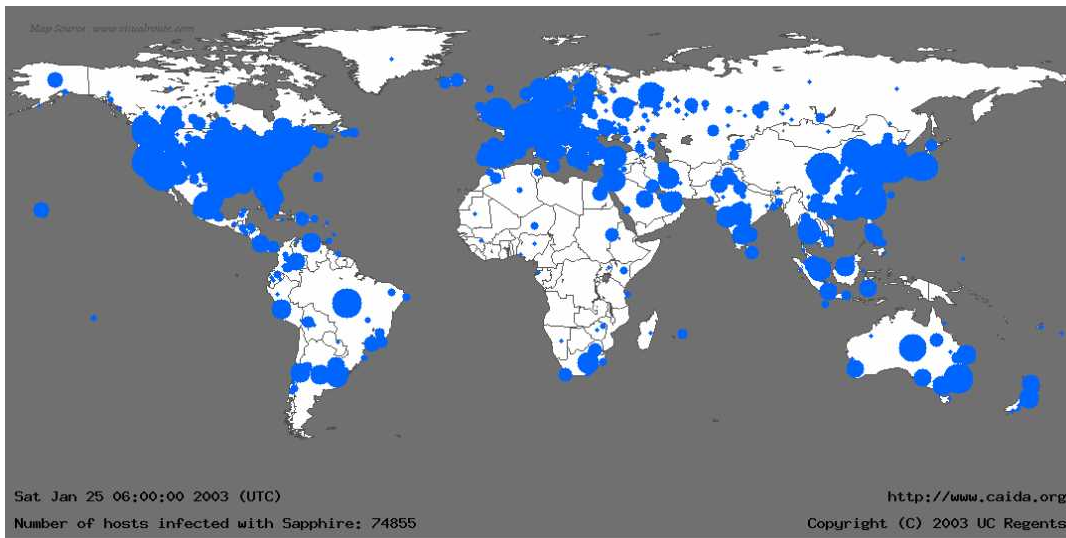


Figure 1. Slammer Global Infection

Mi2g, a London based market intelligence firm, calculated that the Slammer worm caused “between \$950 Million and \$1.2 Billion in lost productivity in its first five days” of operation worldwide [Lem03]. Even so, Slammer was just another in a line of costly malicious actors—the estimated costs of Code Red, the LoveLetter virus, and the Klaz virus were \$2.6 Billion, \$8.8 Billion, and \$9.0 Billion, respectively. These costs include damage directly caused by the malicious code and the administrative costs to correct the infected systems, including the initial cleansing and repair of the systems. Companies must now maintain a constant vigil against malicious actors by keeping staff updated and equipment protected against future attacks. However, the largest financial impact is the loss of the ability to conduct business [Lem03].

The large financial impact of these various malicious logics show organizations need to analyze their operation and capabilities to detect and even prevent future attacks. Researchers and the computing community must understand the operation of malicious logic to provide an effective detection, prevention, and response to future attacks.

## **1.2 Overview and Goals**

This research determines through the use of mathematical simulation and live analysis of malicious code on an infected system, whether the Slammer based routing worm proposed by Zou is faster than the previously observed Slammer worm [ZTG05]. This research also determines through those same simulations and analyses whether the Single Slash Eight (SSE) routing worm proposed in this research is faster than the Slammer based routing worm proposed by Zou [ZTG05]. A third goal of this research is to analyze the statistical randomness of the IP addresses generated by the Slammer worm to establish a basis for the use of random number generators in the mathematical models used to generate infection rates. As part of the validation effort, the Matlab mathematical model simulation is compared to previously observed Slammer data. The final goal of this research is to compare the various 2003 scanning worm infection, the year Slammer was originally released, to the speed that could be possible on computing systems of today.

## **1.3 Thesis Overview**

Chapter 1 provides a brief synopsis of the motivation for this research, an overview of the experiment, the goals of the research, and an overview of the thesis structure. Chapter 2 covers the basics of malicious logic operation with particular

emphasis on worms and delves into the previous research conducted in the area of worms. Modeling of the Internet, Internet protocol addressing, and the concept of the continual increase of computing system speed are also addressed.

A detailed explanation of the experimental design is provided in Chapter 3 including the problem definition with the goals and hypothesis, the experimental approach, and the assumptions and limitations for this research. Chapter 3 also discusses the system boundaries and services, the workload for the system, and performance metrics used in this research. The parameters and factors that form the basis of this research are also discussed. Finally, the evaluation technique, experiment design and configuration, and a discussion of the analysis and interpretation for this research are discussed.

The results and analysis of the data collected are presented in Chapter 4. The first section determines the packet generation capabilities of the original Slammer worm. The second section analyzes the randomness of the Slammer worm IP address generation. The third section determines the infection rates of the original Slammer worm and the Slammer routing worm proposed by Zou [ZTG05]. Chapter 4 demonstrates the speed difference between the worms on systems available today compared to that in use during the original Slammer worm outbreak. Chapter 4 ends with the coverage of the Single Slash Eight routing worm developed herein.

Chapter 5 is a short summary of the research problem and conclusions. The contributions and significance to the computing community and recommendations for future research are also contained in Chapter 5.

Throughout this thesis to improve the readability and flow, the research cited directly in the thesis text is referred to by the last name of the first author credited in that research. As such, the research performed and data created by Zou et al [ZTG05] in their research of routing worms will be referred to simply as Zou. The Slammer worm research completed by Wei et al [WMS05] is referred to as Wei and so on. There are two Slammer analyses cited in this research that were completed by Moore et al [MPS03] [MPW03] where the same authors wrote both documents. Both of these are referred within the text by simply Moore however, the end citation specifically names which research the information came from.

#### **1.4 Summary**

This chapter provides a brief synopsis of the motivation for this research with an introduction to the significant impact caused by malicious logic. The goals, with an overview, of the experiment for this thesis was provided in Section 1.2. Finally, an overview of the structure of this document was presented in Section 1.3.

## II. Literature Review

### 2.1 Introduction

This chapter presents the fundamentals of worm propagation and recent research into worm propagation and modeling. Section 2.2 discusses the basics of computer worms and their operation when released onto a network, as well as a more detailed analysis of the routing worms being assessed in the experiment. Section 2.3 investigates the difficult process of accurately simulating the Internet. The discussion of the division of Internet addresses is given in Section 2.4. Within Section 2.5, the concept of Moore's law as it relates to the increasing speed of computing systems is examined. Section 2.6 discusses previous worm propagation modeling, results, and limitations. Finally, Section 2.7 summarizes all of the previous sections.

### 2.2 Worms

This research considers the operation of random address scanning computer worms across an IPv4 network. One of the major reasons computer worms work so well is that Microsoft Windows, the operating system used throughout the world, has a market share of 94% [Fes04]. This homogeneity, or genetically similar software makeup, has both negative consequences and provides the consumer great benefits. The benefits include lower cost products, easier portability and increased services. The consequences associated with this homogeneity are the ease with which a malicious logic program can move from system to system.

The cost of malicious logic to consumers, companies, and the government is the motivation for this research. It is estimated that computer virus attacks caused \$55



billion in damages in 2003 and that sum was on the rise in 2004 [Mar04]. With the costs from the damages skyrocketing and money to prevent and protect against malicious logic becoming a mandatory expenditure, one can easily see that malicious logic is and has been a serious threat to home computer users, businesses, and the Internet community in general.

Dr. Fred Cohen coined the term “computer virus” in 1984 due to the similarity to their biological counterparts plaguing the human race [Sla95]. His research formed the basis for the epidemiological models. If system A can infect system B, and system B can infect system C then system A can infect system C [KeW91]. Since that initial comparison, a virus has been defined as “a set of instructions which, when executed, spreads itself to other, previously unaffected, programs or files” [Hof90]. However, this does not fully define malicious code, as some malicious code requires a user to open an attachment or possibly an email. In these cases the user is not “directly” acting upon the code; they are activating the “trigger” (e.g., the opening of an attachment). A more complete definition of malicious logic is a program that “modifies or destroys data, steals data, allows unauthorized access, exploits or damages a system” [Hei04] and in general “does something that the user did not intend” [Hei04].

Although, computer viruses are not “living” entities with the ability to build up “immunities,” the programmers of malicious logic are becoming better at creating and writing malicious code. This, in effect, makes the viruses more resilient to correction, detection and prevention, and more tenacious in their ability to infect new systems. This problem of evolution makes defending against and defeating malicious code more difficult with every new generation and makes computer viruses similar in that sense to

their biological counterparts [ZTG05]. Given that expectation, each worm's infection rate should be faster than the last; the SSE routing worm should be faster than the Slammer based routing worm which should be faster than the original Slammer worm.

Worms are often confused with viruses. This confusion is due to the merging of malicious code techniques and the blurring of the lines between application operations. Worms, unlike viruses, have tended in the past to not directly harm the system they are on. Worms replicated in the background and most computers continued to operate. The Witty worm was an exception to this. It contained code that randomly deleted portions of a hard drive attached to the system it was residing on [ShM04]. Even so, all worms are malicious actors. First, worms perform actions not intended by the owner of the system. Worms often increase in size filling up a hard drive, perform data mining, and can bring Internet communication to a standstill through the flooding of the Internet with overwhelming amounts of infected packets.

A proper definition of a worm is a form of malicious code, either standalone or file infecting, that acts with or without human intervention and spreads across a network [KiE03]. This simple definition combines all the aspects of a worm while separating it from a virus or a Trojan horse. A Trojan Horse is a set of malicious code that is hidden within another program, much like the mythical Trojan Horse of the Greeks. Viruses differ from worms in that, a worm does not attach directly to another object or program; worms are standalone code. Worms, unlike Trojan horses, replicate for further infections [KiE03].

Worms can be classified into three groups based on how they operate: E-mail (or client application) worms, Windows file sharing worms, and traditional worms [KiE03].

E-mail worms, as the name suggests, exploit weaknesses in an e-mail system or in application software to propagate (e.g., Melissa [KiE03] used e-mail, and Bilbrog [KiE03] used Internet Relay Chat). This style of worm usually requires some form of user action, such as opening an e-mail attachment. Windows file sharing worms exploit the various file sharing capabilities (i.e., Server Message Block and Common Internet File System) of Windows that allow small groups to work on the same files (e.g., Nimda and Gaobot worms [KiE03]). Traditional worms attack using standard Internet protocols (e.g., TCP/IP, UDP) and operate autonomously once activated [KiE03]. Within this class of traditional worms are the scanning worms. Code Red, Slammer, Witty, and the Slammer based routing worm proposed by Zou are all types of scanning worms [ZTG05]. They all probe the available IP address space to find and infect vulnerable systems [ZTG05].

### **2.2.1 Pseudo Random Number Generation in Worms**

All scanning worms use some form of Pseudo Random Number Generation (PRNG). As the name implies, a PRNG generates pseudo random numbers for use in various applications. The problem with PRNG is that it is not a truly random process but rather an algorithm that generates a sequence of numbers with little or no discernable pattern present in the sequence [Bla06]. This means that no matter how random the numbers generated appear to be, they are predictable. Thus, if a person knows the number that “seeded” the PRNG they will be able to predict the series of numbers generated by the algorithm [Haa99]. However, for the purpose of generating a varied distribution of numbers to be used as target addresses, the PRNG works well as has been shown by the speed at which the worms studied in this research propagate.

### **2.2.2 Code Red**

Code Red was released on the Internet at 1000 hrs UTC on 13 July 2001 and exploited a vulnerability in the Windows Internet Information Services (IIS) that was discovered almost a month earlier by eEye [MSB02]. Windows IIS is a web server architecture for managing website and application availability. The vulnerability was an error in the Window IIS Indexing Services that allowed a remote intruder to run arbitrary code on the victim system [CER02]. Code Red had an error in its random number generator that limited its ability to scan for IP addresses and so its propagation speed was inadvertently restricted. Code Red version II (hereafter called Code Red as the two versions operated identically other than the random number generator), released six days after Code Red, corrected this coding error in the random number generator and infected systems at an exponential rate. Code Red generated 100 scanning threads; each thread randomly selected an IP address and tried to set up a connection on port 80. Code Red was programmed to scan the IP address space uniformly [MSB02].

An unusual characteristic of the scanning threads was that the 100<sup>th</sup> thread would try to deface the currently infected system's web site if it was an English Windows 2000 system. If the target of the 100<sup>th</sup> thread was not an English Windows 2000 system, the thread would be used to infect other systems rather than trying to deface the web site. Once infected, the system would become a platform for launching new attacks. If the target system was not a web server or it could not be infected, the thread would generate a new random IP address and try again [MSB02].

The Code Red worm could only infect a Windows system with IIS installed. At the time, Microsoft estimated there were six million Windows IIS servers on the Internet.

However, Code Red was programmed with a stop time and did not have the opportunity to infect the entire population. It stopped its propagation at 0000 hrs UTC on 19 July 2001 after infecting an estimated 359,000 computers in less than 14 hours. The cost of this worm was estimated to be in excess of \$2.6 billion [MSB02].

As Code Red was, at the time, one of the most aggressive worms observed, there have been many experiments and research projects analyzing its operation. Code Red was one of the worms used by Zou as a basis for validating the speed of their proposed routing worms. The Zou routing worm research is discussed in more detail in Section 2.2.4 [ZTG05].

### **2.2.3 Slammer**

Slammer, also known as SQL Slammer and Sapphire, did not attack the end system computers (i.e., the personal home computer), but it wreaked havoc by virtually shutting down portions of the Internet as it spread itself among the core servers and throughout Internet [HyE03]. Core servers provide Internet access for multiple computers at a company or small network. Of the 13 Internet root name servers, the servers that form the essence of the domain name system, five were shutdown by Slammer traffic and close to 20% of all data sent across the Internet was lost during the outbreak [HyE03]. Slammer was the fastest spreading worm ever observed [MPS03]. It doubled the number of infected systems every 7.5 – 9.5 seconds in the first minute and managed to infect over 90% of its potential 75,000 victims in about 10 minutes [MPS03].

Slammer operated by exploiting a “buffer overflow” vulnerability in the Microsoft SQL Server 2000 Operating System by gaining access to the computer memory stack and replicating itself. Slammer sent massive amounts of data across the

Internet during its attempt to infect other systems; reaching over 55 million scans per second in just three minutes [MPS03]. The small size of Slammer added to its effectiveness by allowing it to be sent in a short time. This small size provided an additional benefit of initially hiding its existence since the large file transfers prevalent on the Internet masked its presence. Even with the small size of 404 bytes, the amount of data being sent across the Internet by Slammer during the three minute interval was over 23,000 gigabytes every second.

Also aiding Slammer's effectiveness was its use of User Datagram Protocol (UDP) for communication [HyE03]. UDP allows transmission with no requirement to establish a return path acknowledgement. This meant Slammer could scan the Internet without concern for establishing a connection to the targets, thereby further increasing its speed. The consequences could have been much worse; Slammer had a small flaw in its program that limited the number of Internet system addresses it could scan for infection [MPS03]. However, this did not appear to impact the speed at which Slammer was able to spread. This flaw did, however, limit the ability of researchers to calculate the IP address range vulnerable to the Slammer worm, as they had been able to do with previous worms [MPS03].

#### **2.2.4 The Border Gateway Protocol (BGP) and /8 Routing Worms**

The Border Gateway Protocol (BGP) and "/8" routing worms are malicious code proposed by Zou [ZTG05]. These worms are an advanced form of malicious code using two techniques to increase their speed of infecting vulnerable systems and creating an overload of malicious packet traffic on the Internet backbone.

The BGP routing worm is, as the name implies, a scanning of the BGP routing tables for valid computer addresses to attack. The BGP routing worm is named after the protocol used by Internet Service Providers (ISPs) of various tier sizes as their inter-autonomous system routing protocol to exchange information between ISPs. The BGP routers contain ISP IP addresses, and Zou proposed an autonomous worm that harvests valid IP address tables from the BGP routers. Thus, this makes the BGP routing worm a more precise Internet scanner and should be at least three times faster than any previous worm [ZTG05]. The second capability of a BGP routing worm is the ability to attack, say, only a specific country, company, Internet service provider, due to the inherent geographical information in routing tables [ZTG05].

The “/8” routing worm implements similar techniques with one difference from the BGP routing worm. Instead of a large block of code to query the BGP routers for their prefixes to hone the worm’s search, the “/8” routing worm is pre-coded with the 116 IPv4 “/8” routable addresses. This reduces the amount of code required and therefore the size compared to the BGP routing worm. Inserting the 116 IPv4 “/8” routable address prefixes would only increase Slammer’s size by 116 bytes to 520 bytes [ZTG05]. The code to perform this modification was not available in published research.

Due to the public availability of the BGP routing tables, developing an effective worm is comparatively easy because the tables provide a known good range of Internet protocol addresses to attack which make the worm spread more effectively by reducing the required scanning space without the risk of missing a target. As of September 2003, over 28% of the IPv4 addresses were BGP routable, reducing the required scanning space by almost 70% [ZTG05].

In addition to spreading quicker and being able to target a specific region or range of IP addresses for attack, there are two other challenges that make the worms problematic for the Internet community. First, the BGP and “/8” routing worm can cause even more congestion than Slammer because the IP addresses generated by the routing worms are inherently BGP routable. Unlike the other traditional scanning worms including Slammer which use TCP/UDP that can be easily dropped if the generated address is invalid, the addresses generated by a BGP worm are always valid at the BGP router and therefore forwarded. Since Slammer and other worms scan the entire IPv4 address space 70% of their IP addresses generated are non-routable and dropped [ZTG05]. Therefore, a majority of the traffic generated by Slammer, did not even appear on the Internet backbone and did not cause any congestion. Despite this fact, Slammer caused severe congestion to several local area networks and the Internet as a whole.

The second challenge presented by the BGP and “/8” routing worms is they are more difficult to detect compared to previously observed scanning worms (e.g. Code Red, Slammer, Witty). One of the major identifiers used to track scanning worms is the large amount of traffic generated without response. Because the BGP and “/8” routing worms generate packets that will be dropped by the routers and furthermore since the failed responses would be delayed due to the routability of the infection packets, detecting the worm based on an illegal traffic method would be slow [ZTG05].

### **2.2.5 The BGP Routing Worm Propagation**

The BGP and “/8” routing worms were based on a mathematical model that reflects the actual propagation parametrics observed in previous worms. The infection



rate of the BGP and /8 routing worms were modeled using a uniform-scan worm model described as [ZTG05]

$$\frac{dI_t}{dt} = \frac{\eta}{\Omega} I_t (N - I_t) \quad (1)$$

where  $I_t$  is the number of hosts infected at time  $t$ ,  $N$  is the number of vulnerable systems,  $\eta$  is the scanning rate, and  $\Omega$  is the address space requiring scanning. Using Code Red as a basis, the BGP and /8 routing worm infect rates versus the observed Code Red rates are shown in Figure 2.

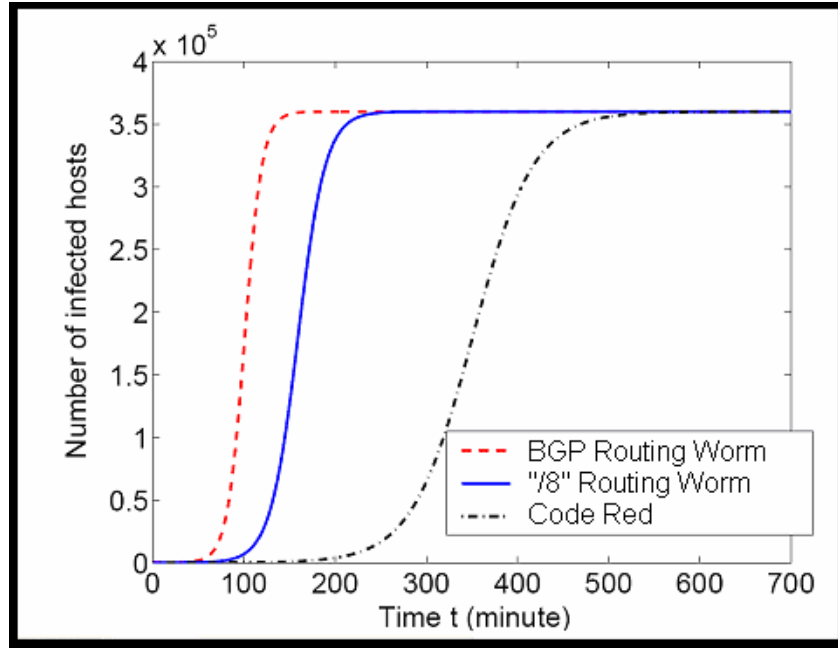


Figure 2. Zou Code Red Worm Simulation

The values used for the comparison were  $\eta = 358$  scans per minute,  $N = 360,000$ , and  $I_t = 10$  systems infected. The address space to scan,  $\Omega$ , was set to 4,294,967,296 for the Code Red infection curve and 1.95 billion for the 116 Internet Assigned Numbers Authority (IANA) “/8” BGP routable addresses for the “/8” routing worm. The BGP routing worm had all of the same variables except for the available address space. For

the BGP routing worm  $\Omega$  was set to 1,228,360,647 to reflect the further refinement of the scanned address space. The outcome of this experiment showed that both the BGP and /8 worms operated significantly faster; more than 3 times as fast as the Code Red worm [ZTG05]. Unlike the Zou Slammer-based worms (discussed below), there were no changes in the scan rate made in the Zou Code Red-based Worms due to the the increase of code size required to make the routing worms [ZTG05].

The model of Slammer as a “/8” routing worm was developed using the uniform-scan worm model with  $\lambda = 4000$  scans per minute,  $N = 100,000$ , and  $I_t = 10$  systems infected [ZTG05]. The address space to scan,  $\Omega$ , was set to 1,946,156,941 addresses. The Slammer “/8” routing worm, referred to as the “routing Slammer worm,” used the same values for  $N$  and  $I_t$ . However, to reflect the reduced address space of a “/8” routing worm and the increased code size of 520 bytes to incorporate the “/8” routing worm code into Slammer,  $\Omega$  was set to 1.95 billion and  $\lambda$  was set to 3108.

A complete list of the variables used by Zou is provided in Table 1. The data revealed that the new “routing Slammer worm” (hereafter called the Zou Slammer routing worm) was more than twice as fast as the original Slammer infection rate as shown in Figure 3 [ZTG05].

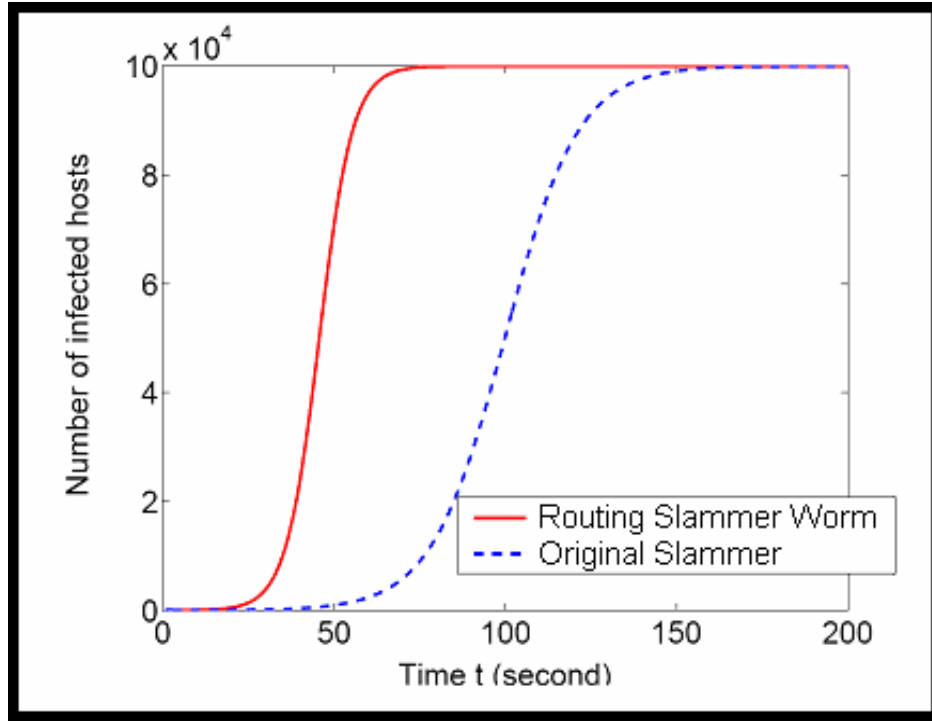


Figure 3. Zou Slammer Worm Simulation

Table 1. Zou Worm Simulation Variables

	Number of IP addresses	Number of Vulnerable Systems	Packet Generation Speed	Initial Number of Infected Systems
Code Red Worm	4,294,967,296	360,000	358 per Minute	Ten
Zou Code Red "/8" Routing Worm	1,946,156,941	360,000	358 per Minute	Ten
Zou Code Red BGP Worm	1,228,360,647	360,000	358 per Minute	Ten
Zou Slammer Worm	4,294,967,296	100,000	4,000 per Second	Ten
Zou Slammer Routing Worm	1,946,156,941	100,000	3,108 per Second	Ten

### 2.3 Internet Modeling

Due to the Internet's complexity, rapid growth, and constant change, any attempt at modeling it will be severely limited. Mathematical models, software simulations, and

hardware simulations all suffer from a lack of size or ability to effectively mimic the operation of the Internet. The biggest problem any model of the Internet is its inability to model the true vastness of the global Internet community [FIP01]. There were an estimated 16 million users in December of 1995 [Gro03], [IWS07]. The number of users has increased to 1,093 million by December of 2006 [IWS07]. Even the most advanced computer simulation software falls far short of being able to represent that many nodes on a network.

Another problem with simulating the Internet is heterogeneity. The Internet, while dominated by Microsoft products, is not a network of similar computers. There are differences in platforms, link sizes, data rates, and network topologies that increase the difficulty in accurately simulating the Internet even further. Also confusing the issue are the differences in protocols used by these varied systems and networks. Each of these protocols has their own operating characteristics, formats, and traffic loads to consider to generate an accurate model [FIP01].

A third area of concern is the difficulty in representing the amount of traffic load. The Internet has congestion control techniques and dynamic routing capability. Simulating this adds significant complexity to a model. Further, Internet traffic is not constant. The high traffic pattern for each network follow the same pattern as an average workday; the traffic tends to increase as the workday begins, tails off over the lunch period, starts to climb again after lunch until the end of the workday, and finally exhibits some increase in the evening hours after dinner that has been attributed to home/recreational computer use [FIP01]. Figure 4, shows a typical traffic pattern of

several states in the United States that exhibit this pattern for a rise as the workday begins and a falloff of traffic as nightfall occurs [Whe02].

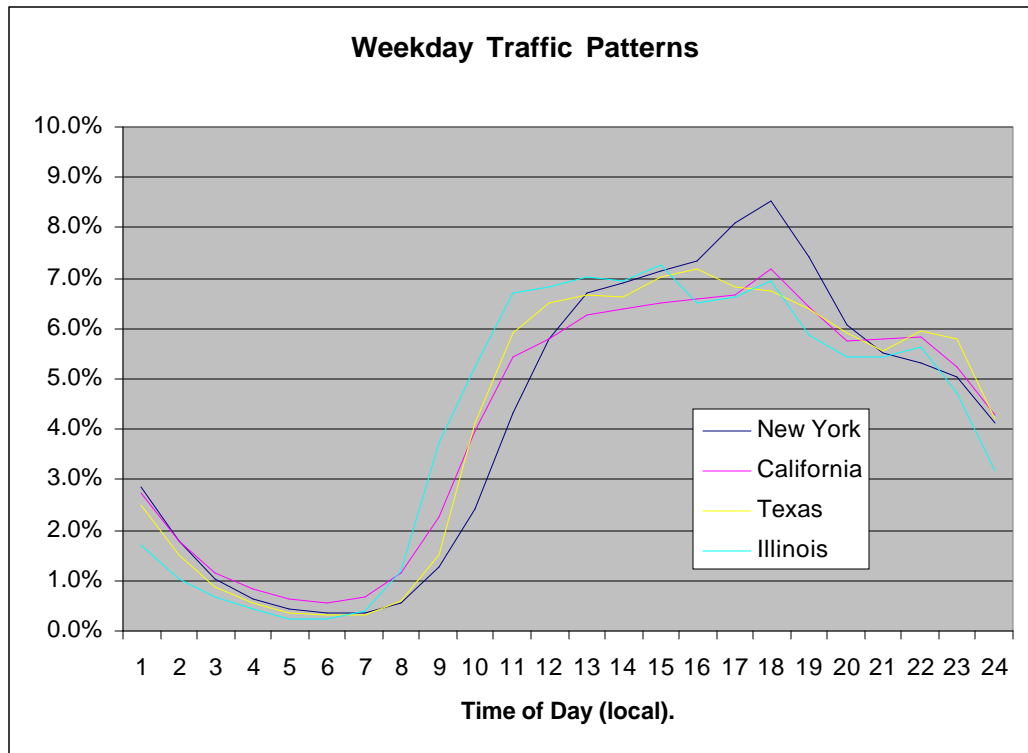


Figure 4. Pattern of Daily Network Traffic

The final factor presented by Floyd and Paxson is that a model of the Internet will likely not be useful tomorrow. The Internet changes everyday in its operation, size, and use. Some of the areas that make the future Internet difficult to predict are pricing structures, the explosive growth of wireless, and the currently undeveloped new “killer application” [FIP01].

## 2.4 Internet Protocol Addressing

IPv4 is the predominant protocol used for Internet communication today, but on the horizon is the IPv6. IPv4 has an address space of  $2^{32}$  yielding 4,294,967,296 IP addresses. However, due to the reserved IP ranges and limitations on the range usable by

Internet community, there are just over 3.5 billion useable IP addresses [WaC02]. Sixty-five percent of the IPv4 address space is assigned to the United States and the growth of the Internet in Europe and Asia is causing problems with a lack of available address space [Huf03].

Internet Assigned Numbers Authority (IANA) assigns and manages the available Internet address for the world. From the IANA the Regional Internet Registries (RIRs) receive large groups of IP addresses. These RIRs manage their assigned smaller addresses and distribute an even smaller range to the large Internet Service Providers (ISPs) for their area of responsibility. In turn, the large ISPs provide Internet connection service to the smaller local ISPs and end users.

Classless Inter-Domain Routing (CIDR) is a prefix based standard for the interpretation of IP address groupings to allow easier use and discussion. The slash number designation, such as “/8,” denotes the prefix aggregation of the IP address from the full 32 bit IPv4 address. Thus, a “/8” takes the 32 bit IPv4 address from approximately 4.3 billion possible addresses and reduces the address to the last 24 bits as the first 8 bits are masked. This reduces the IPv4 address space under consideration to 16,777,216. This “/8” grouping is the typical CIDR size provided to the RIRs for dispersal to the ISPs.

The “/8” routing worm reduces the overall IPv4 address space by instituting a CIDR “/8” allocation table as part of its functionality [ZTG05]. Therefore, out of the possible 256 “/8” address groups, only 116 are IANA IP routable addresses. Based on these restrictions and reserved addresses dedicated under the authority of IANA, the scanning required by a “/8” routing worm is reduced to 45.3% of the IPv4 address space

[ZTG05]. This means that instead of scanning the entire 4.3 billion IPv4 address space as Slammer did, a CIDR “/8” based routing worm would only have to scan 1.95 billion addresses.

## **2.5 Increasing Speed of Technology**

The speed of both the Internet and computers is increasing each year. According to the often-quoted Moore’s Law, the speed of computer processing power roughly doubles every 18 months. Thus, if one uses Moore’s Law to compute the difference in a computer attached to the Internet between 2003 and 2006, the system in 2006 should be about 4 times faster. Moore’s Law has been used to predict everything from disk storage capacity to digital camera resolution. Others have estimated the increase of network capacity used Moore’s Law.

In fact, the growth of the Internet has more than matched the estimates of Moore’s Law. During 1996-2002, the traffic on the Internet backbones in the United States doubled every year and the infrastructure kept pace with this exponential growth [Odl03]. As previously mentioned, the number of users increased 68 times in just eleven years from 16 million in 1995 to 1,093 million in 2006. This does not follow the Moore’s Law rate of expansion.

## **2.6 Related Research**

The primary focus of this research is to analyze the performance of the original Slammer worm, the Slammer routing worm as proposed by Zou, and the SSE routing proposed in this research on an IPv4 network [ZTG05]. The Slammer worm has been used in many worm studies where a mathematical model was developed to simulate its

effects [WMS05] [PeS04] [YuW04]. The BGP and “/8” routing worms are worms proposed by Zou and modeled mathematically to compare against the observations of Slammer and Code Red [ZTG05].

The Directed-Graph Epidemiological (DGE) model has been the basis for many worm models [KeW91]. It extended the standard epidemiological model to a directed-graph model and uses in the analysis and simulation of viruses. DGE applied the simple SIS (Susceptible -> Infected -> Susceptible) epidemiological model to various graphs to emulate the propagation of viruses [KeW91]. This was one of the first attempts at mathematically adapting virus propagation to the epidemiological model [KeW91] and became a reference for future mathematical malicious logic modeling [CGK03] [KRD04] [RSL04] [YuW04].

Chen used a mathematical model of Code Red to study the propagation characteristics of worms [CGK03]. It was claimed that the model of Code Red closely matched the infection rate curve data collected from the original Internet introduction of Code Red, however no statistical basis for the claim was provided. The model provided a formula for detecting, monitoring, and defending against further worm attacks and provided a way to understand worms and aid in the defense against them in the future [CGK03].

Permulla and Sundaragopalan extended worm modeling to the packet-level. Their research developed a high-fidelity packet-level network simulation that emulated the operation of worms on the Internet and included some real operational subsystems. They included experiments with simulated live monitoring and defensive systems.



However, the monitoring and defense system was not included in the mathematical model of worm propagation [PeS04].

Wei Yu considered four classes of worms and characterized their operation through modeling and numerical analysis. Yu analyzed pure random-based, peer-to-peer hit list-based, cooperation-based, and non-cooperation based strategies for worm operation. The numerical analysis found the most effective worms used hit lists [YuW04]. This is similar to Zou when considering the “/8” routing worm as a form of “hit list” worm [ZTG05]. An epidemic model simulated the actions of worms on the Internet [YuW04].

The goal of the research performed by Joshua Hansen was to prove that a worm could be programmed to find exploits on a heterogeneous network and use those exploits to propagate without interaction with its creator. Hansen defines a heterogeneous network environment as a network that has different operating systems; however, these systems have a common software program that could be exploited [Han03]. This is consistent with current Internet topology since, in 2004 Microsoft had the clear majority of operating systems on the Internet [Fes04]. Java was chosen as the basis for the Hansen exploit because of the wide use of Java across many operating systems including Microsoft and UNIX. A six-node network using IPv4 with three real and three simulated nodes using a virtual machine was used for this simulation. Of those nodes, the experiment included one real and one virtual node, which communicated wirelessly. Hansen did not use a live worm, but hard coded the exploit parameters into the modeling code and initially programmed an artificial vulnerability into each node [Han03].

Wei used Slammer data to validate a packet-level worm simulation on the Emulab test bed using a realistic background traffic model [WMS05]. Their initial research used the original Slammer data [MPW03], which consisted of 75,000 vulnerable systems and an average scan rate of 4,000 packets per second and mathematical models of worm propagation. The worm propagation models were built using the variables of vulnerability ratio, scanning rate, infection delay, and scanning strategy. The scanning strategy took into account how much of the address space the worm scanned and how the worm performed the scan. It also modeled network congestion and network failures. The resulting infection rate curves, shown in Figure 5, include an estimate of how quickly a set of vulnerable system would be infected without network congestion [WMS05].

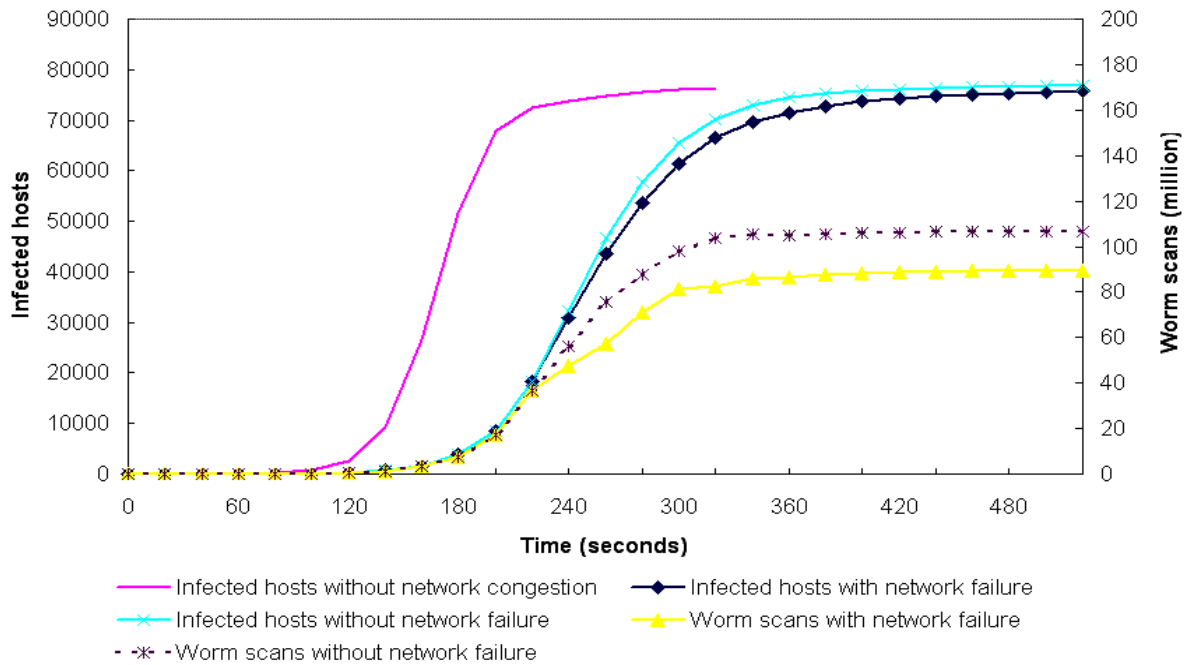


Figure 5. Wei Slammer Worm Simulation

Figure 6 shows Slammer worm propagation with the constant and random background traffic values. Wei observed that there was a negligible difference between

the infect rate of Slammer in the exponential and constant background traffic experiments and attributed this to Slammer using UDP packets that could easily overcome the model of the background traffic and congestion. A worm based on TCP would likely exhibit a larger sensitivity to the background traffic values [WMS05].

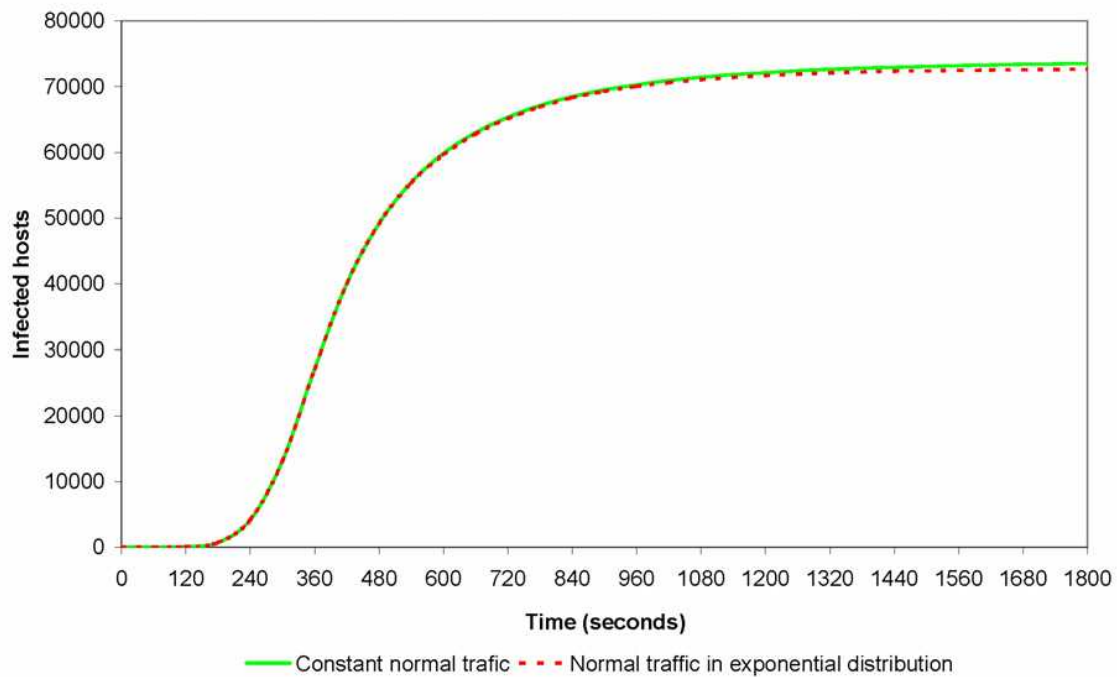


Figure 6. Wei Slammer Worm propagation with Network Congestion

Wagner also used Slammer to validate models of network and bandwidth latency constraints. The simulator was based on observed speed and connection data gathered from the peer-to-peer file sharing software Napster and Gnutella hosts. During the Slammer portion of their research, Wagner created a 10-group network, set the initial number of infected systems at 100, and the vulnerable systems at 75,000 [WPD03]. The initial number of infected hosts appears to be arbitrarily assigned. The research found the collected data matched the infection doubling rate observed during the original release of the Slammer worm and the overall scanning rate after three minutes [MPW03], but

deviated significantly from the observed propagation speed. This deviation was attributed to an increase in line speeds since Slammer was originally released [WMS05].

## **2.7 Summary**

This chapter provides an overview of related research. Specifically, the operation of malicious logic with an emphasis on worms is provided. More detailed coverage of the Code Red, Slammer and routing worms and their previously observed propagation characteristics is also provided. Each of the worms is a version of a scanning worm that is programmed to randomly scan the available address space.

There have been many mathematical simulations and models of both Slammer and other scanning worms on IPv4 networks. Only Perumalla, Sundaragopalan and Hansen tried to include actual operational systems into their research; however they did not use the actual worms [PeS04] [Han03]. Wei performed some detailed simulation of worm propagation using mathematical models and a network of computers [WMS05]. Zou proposed and mathematically evaluated a pair of routing worms that proved to be faster than both the Code Red worm and the Slammer worm [ZTG05]. None of the previous research has involved re-releasing the Slammer worm onto an actual network to observe its effects for analysis. Rather, they relied on matching the previously observed data to build and validate their models.

## III. Methodology

### 3.1 Introduction

This chapter describes the methodology used to create the experiment, the trials to test the hypotheses, and the data analysis. Specifically, Section 3.2 present the problem definition including the goals and hypotheses, the approach and the experimental assumptions and limitations. Sections 3.3 and 3.4 cover the system boundaries and system services. The workload, consisting of the various worms tested in this research, is discussed in Section 3.5. Section 3.6 covers the metrics collected. The parameters and factors for this experiment are in Sections 3.7 and 3.8. The techniques used for evaluating the data collected in this research are covered in Section 3.9. The network configuration and the design of each worm tested is defined in Section 3.10. Finally, Section 3.11 presents the analysis and interpretation of the results.

### 3.2 Problem Definition

#### 3.2.1 Goals and Hypothesis

The primary goal of this research is to characterize the ability of the original Slammer worm, the Slammer based routing worm proposed by Zou, and a new Single Slash Eight (SSE) routing worm proposed by this research to infect vulnerable systems within a given address space on an IPv4 network. The infection rate of these routing worms across an IPv4 network are determined and their operation compared to the original Slammer worm. This research also investigates the Slammer worm's ability to generate a uniform random IP addresses in a given address space. Finally, the

implications of the speed increase of computing systems available today versus those in use during the original Slammer release is discussed.

Since the Slammer routing worm has only been studied through mathematical models [ZTG05], this research uses observed Slammer characteristics to mimic the expected operation of a Slammer routing worm by analyzing its ability to generate random IP addresses. To evaluate the randomness of the Slammer IP address operation, each IP address generated will be evaluated as a whole and individually by octet. The propagation speed of original Slammer worm, the Slammer routing worm, and the SSE routing worms are compared assuming a computing system from 2003 and a system in 2007. This is accomplished using data collected from a modern infected system for the 2007 data and the use of the original Slammer characteristics for the 2003 data. The experiment determines whether there is any significant difference between the propagation speeds in 2003 versus 2007.

### **3.2.2 Approach**

The original Slammer worm code is sent to a vulnerable system, also called the victim, by a carrier workstation. Once the infection packet is received at the victim, the worm code executes and the worm generates packets autonomously for further infections. These infection packets generated by the victim are collected and used to characterize the operation of the Slammer worm for simulation. These measured propagation characteristics are combined with mathematical simulations and compared to the known statistics of the Slammer worm in the wild.

The IP addresses of the Slammer infection packets are analyzed to determine the worm's ability to generate uniform random addresses. To ensure uniform distribution

within the IP addresses, each of the last three octets is evaluated individually to determine whether they are each uniformly random in distribution within their smaller address space. Because the first octet is being simulated for the various routing worms, it can be represented by any valid octet value and is not germane to this experiment.

Once the randomness has been determined, a mathematical simulation of infection rate is generated using the variables required for original Slammer, the Slammer routing worm, and the SSE routing worm. Within the mathematical simulation, the size of the IP addresses available for scanning, the initial number of infected systems, and the number of vulnerable systems are controlled and modified based on the particular worm being evaluated.

Finally, the propagation characteristics collected from the Slammer infected system are evaluated for their generation speed. This data is compared to the original Slammer characteristics for the 2003 computing systems versus the current computing systems available in 2007.

### **3.2.3 Assumptions and Limitations**

It is assumed that a future exploits allowing the propagation of a worm like Slammer will continue to occur. The primary limitation of this experiment is the inability to accurately represent the Internet architecture as mentioned in Chapter 2. Without the ability to represent the Internet or a large network, the true propagation characteristics of the worms is necessarily limited.

## **3.3 System Boundaries**

The System Under Test (SUT) is an IP network known as the Network Under Attack. As shown in Figure 7, the number of network nodes, protocol, topology, and links between

nodes are all components of the Network Under Attack. The specific Component Under Test (CUT) is the Slammer worm, Slammer routing worm, and SSE routing worm.

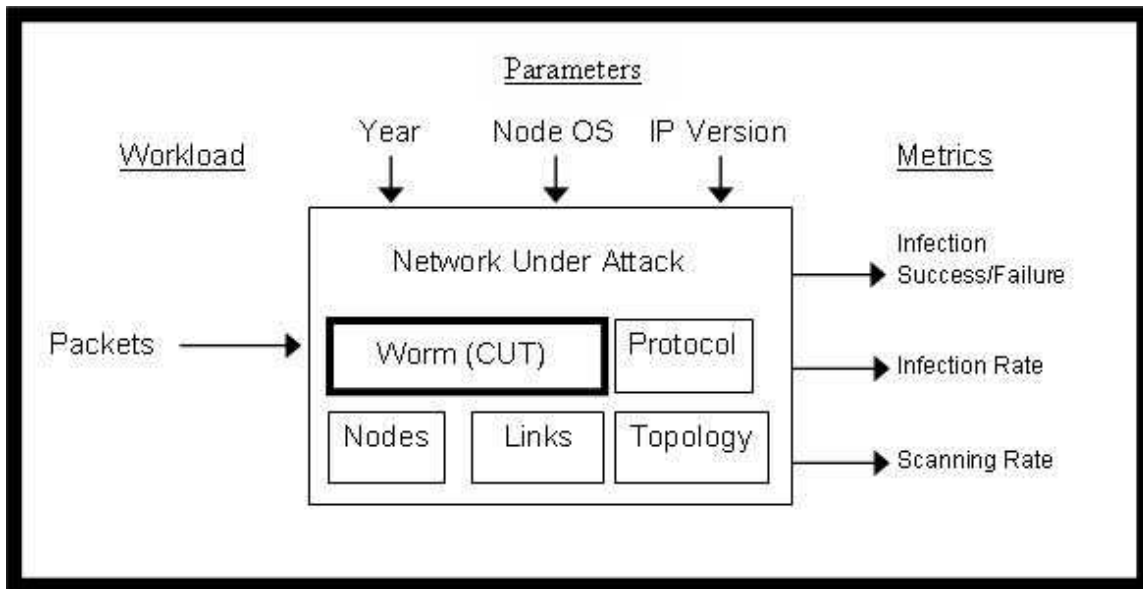


Figure 7. Network Under Attack

### 3.4 System Services

The service provided by this system is the transport of user data between nodes. The outcome of this service is success, failure, or degraded operation due to worm infection. Failure could be due to worm infection preventing valid user data from crossing the network.

An illegitimate but interesting use of the system service is the propagation of worms. The outcome of this service is also success, failure, or degradation. Success is defined as the transfer of a worm from one node to another. Failure occurs when a worm cannot move from one node to another. Degradation could be caused by the reduction of the worm's capability to spread due to the large traffic load.



### **3.5 Workload**

The workload of the system is the worms. The first worm, Slammer, provides a basis for validation on the IPv4 network. The Slammer routing worm is compared to the results generated by Slammer to determine whether it is faster and to [ZTG05] for validation. Lastly, the SSE routing worm is compared against both Slammer and the Slammer routing worm to determine how much faster it is.

### **3.6 Performance Metrics**

The primary metric used in this experiment is the number of systems infected per second. From the Matlab model of infection rate, the infection doubling rate can be calculated. This rate is compared to the observed data to validate the statistical model. The known infection rate of Slammer on the Internet also provides a method to determine whether the Slammer routing worm and SSE routing worm spread faster than Slammer. The scanning rate is measured by the packets generated per second by an infected system for a given time period. This metric is also used to compare scan rates between the systems available for infection during the original Slammer release and those in use today.

### **3.7 Parameters**

#### **3.7.1 System Parameters**

- Number of Nodes: The number of nodes in this experiment is limited by a constant generation of a multicast address in the first octet that could not be resolved during this research effort. Therefore, only one vulnerable node is available for use in the analysis of each worm.

- Link Data Rate: As the congestion on a link increases due to network traffic, the data rate of that link and how much traffic it can carry impacts the spread of the worms. The link data rate varies from link to link and cannot be adequately simulated in this research with the resources available. Therefore, the variation in link capability is not a variable for consideration in this experiment and each link is a direct connection through a switch using Category 5e ethernet cable.
- Operating System: Worms normally target one particular operating system. If the nodes of the network are not running the target operating system, the worm will not spread. Consideration of how non-vulnerable nodes affect propagation is not considered. The target host is always loaded with the vulnerable software component of Microsoft Server 2000, which the Slammer worm exploits.
- IP Version: IPv4 is used throughout this research. This facilitates the baseline comparison of the worms under test against the known propagation characteristics of Slammer.
- Year of the Computing System: There are two time frames considered in this experiment; the year Slammer was originally released (2003) and year that this experiment takes place (2007). These two time frames provide a comparison of the difference in speed of worms released onto systems used in 2003 to those in use today. The use of the average Slammer worm scan rate observed in 2003 and the operation of the Slammer worm on a

computing system available today provides the baseline for this comparison.

### **3.7.2 Workload Parameters**

- Worms: The worms represent a malicious packet workload for the network to transport. Each worm, the original Slammer, the Slammer routing worm, and the SSE routing worm, are all workload parameters in this experiment.

## **3.8 Factors**

The factors is the workload and the year of the computing system. There are three main worms considered in this experiment – the original Slammer worm, the Slammer routing worm, and the SSE routing worm. The original Slammer worm is used as a baseline for comparison and validation of the other worms and mathematical simulation using the archived data and experimental data from other experiments. How the capabilities of the Slammer routing worm and the SSE routing worms compare to the original Slammer worm is the focus of this experiment.

The second set of factors is the difference in packet generation rate between the years 2003 and 2007. The average Slammer worm packet generation rate from 2003 is used to establish the baseline for comparison to the data observed on a computing system of today. To evaluate how a fast the Slammer worm, the Slammer routing worm and the SSE routing worm could propagate on a computing system of today, the actual observations of the scan rate of the Slammer code on the test network is used.

### **3.9 Evaluation Technique**

This experiment directly measures the generation of infection packets on a network. This data and the originally observed Slammer scan rate is provided as the input variable of the packets generated per second to a mathematical model to provide infection rate data.

Two laptops connected by a switch, both described in Appendix A, are used to simulate an attacking system and a vulnerable host. The attacking system sends the infectious worm packet to the vulnerable host via UDP packet to infect that system. Once the vulnerable host receives the infection packet, that host becomes infected and begins to autonomously propagate the worm to randomly generated IP addresses.

The validation of the worm operation and infection rate is accomplished by comparing previously collected Slammer data and experiments from [MPS03], [WMS05], and [ZTG05]. The randomness of the IP addresses is validated through the use of regression analysis and comparison to a statistically generated uniform distribution of the available address space.

### **3.10 Experimental Design**

#### **3.10.1 Network Configuration and System Infection Procedure**

The network configuration of the hardware for each run of the experiment uses the same components. Each run consists of two laptops connected to a switch as shown in Figure 8 connected by category 5e ethernet cable. The specifications for each component are provided in Appendix A.

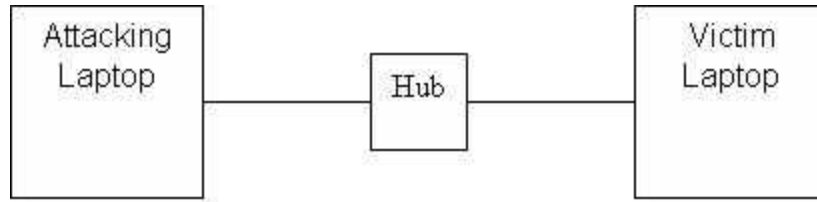


Figure 8. Network Configuration

A detailed description of the Slammer code used for infecting the victim machine is provided in Appendix B with analysis of the assembly code and a diagram of stack operation. The Slammer worm code is sent via a UDP packet to the victim machine using the Netcat tool. Netcat wraps the Slammer binary code in a UDP packet and sends the completed packet to the destination address and port specified. This is accomplished by invoking the Netcat software through the use of the command line as shown in Figure 9.

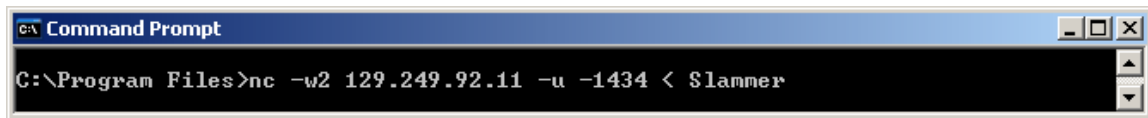


Figure 9. Netcat Infection Command

Each new infection is initiated using this same command line interface. The breakdown of the command is shown in Table 2.

Table 2. Netcat Infection Command Description

nc	-w2	129.249.92.11	-u -1434	< Slammer
Calls Netcat Program	Wait two seconds to close connection	Target IP Address	Open a UDP connection to port 1434	Send the "Slammer" file to the Target

### 3.10.2 Experimental Exploration of Slammer Randomness

To establish a baseline for comparison, the Slammer packets are collected and used to build a statistical graph of the infection rate for comparison to the real world

observed operation of Slammer. The first step in this process is to establish that the IP addresses generated by Slammer are statistically random. The problem with the random number generator in Slammer is well documented in [MPS03] and [MPW03]. This flaw affects the first octet and is therefore not part of the experimental consideration in the generation of random IP addresses. Only the last three octets are analyzed and used for the examination of randomness.

The uniform distribution of the random addresses generated by Slammer is verified by analyzing the infection packets generated by Slammer as an entire IP address space of three octets and individually as to the randomness of the octets themselves. To verify the statistical uniformity of the random numbers generated by Slammer, the total address space under consideration is divided into an evenly distributed groupings based on the number of samples taken. The grouping is established by first taking the upper value range minus one to account for the value of zero to establish the upper range of the data set under evaluation. This upper range is then divided by the number of samples taken which provides the statistical interval for each sample point. The statistical interval is then multiplied by the sample event number to generate the statistical average for that sample point. The sample event number and the statistical average are plotted on an “XY” scatter plot to generate a regression line for analysis of the data collected.

For example, if the experiment called for a statistical spread of a single hexadecimal number with 20 samples the calculations would be as shown in Table 3. To find the upper limit, the total possible values of single hexadecimal number, 16, is reduced by one to account for the value of zero. This upper limit of 15 is divided by the number of samples taken of 20, which calculates the statistical interval of 0.75. The

statistical interval of 0.75 is then multiplied by each sample event number including zero to generate the statistical average for the data set.

Table 3. Statistical Model Example

Samples Taken	20										
Statistical Interval	0.75										
Event No.	0	1	2	3	4	5	6	7	8	9	10
Statistical Average	0	0.75	1.5	2.25	3	3.75	4.5	5.25	6	6.75	7.5
Event No.	11	12	13	14	15	16	17	18	19	20	
Statistical Average	8.25	9	9.75	10.5	11.25	12	12.75	13.5	14.25	15	

This data is then input into an “XY” scatter plot to show the relationship between the samples collected and values expected as shown in Figure 10. The linearity of the data collected is compared to the statistical model to establish whether the data exhibits a similar linearity.

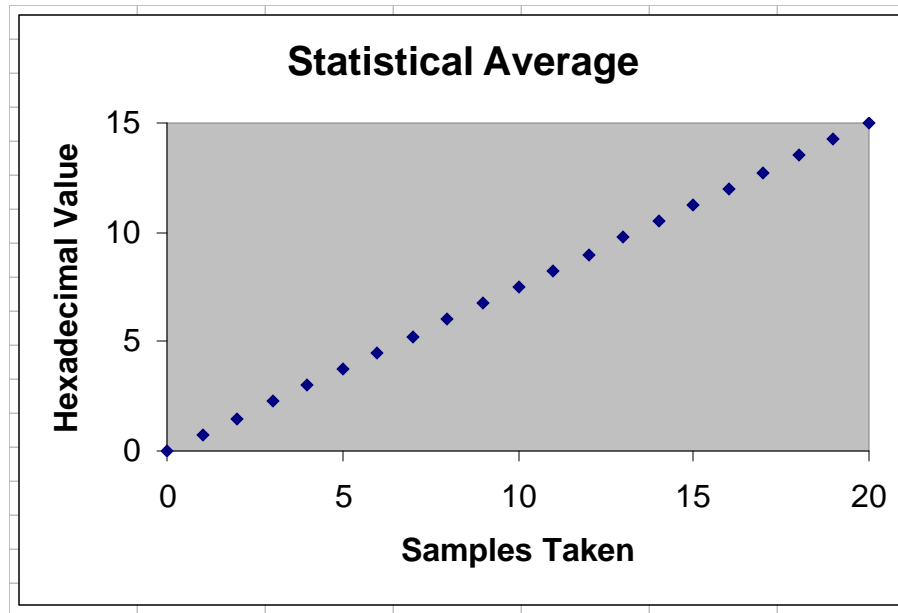


Figure 10. Statistical Model Graph Example

The statistical similarity of the model to the data collected will prove whether the data collected is uniformly distributed and therefore statistically random across the available address space. Table 4 provides sample points of the statistical model used for the single octet and combined octet experiments.

Table 4. Experiment Statistical Model

Address Space	Range of Samples	Samples Taken	Statistical Interval	Example Event Value						
				16	128	192	1024	4096	16384	65535
Single Octets	256	65535	0.00389105	0.062	0.498	0.747	3.9844	15.9377	63.751	255
Three Octets	16777216	65535	256.003891	4096	32768	49153	262148	1048592	4194368	16777215

### 3.10.3 Matlab Model of Infection Rate

The Matlab model used in this research produces a worm infection rate based on several variables. An overview of the operation of the Matlab model code is provided here and in more detail in Appendix C. The number of initially infected systems, the number of vulnerable systems and the number of possible addresses available for scanning are all entered into the model. The number of vulnerable systems is a range that always starts at 1 and ends at the total number of vulnerable systems. The Matlab model uses a pseudo random number generator to generate a number that represents an infection packet for an IP address of a system for each iteration of the code. The Matlab model generates one of these numbers for each system infected prior to that iteration. These IP address numbers are compared against the total number of remaining vulnerable systems. If the IP address numbers fall within the range of the vulnerable systems then the iteration number is documented and the available vulnerable systems is reduced by one. If the IP address is not within the range of the vulnerable systems then the Matlab model



continues to the next iteration. This continues until the vulnerable systems are reduced to zero, which indicates that all of the vulnerable systems have been infected. The Matlab model then generates the file containing the iteration numbers of when each system was infected, those numbers are entered into an Excel spreadsheet and multiplied by the packet per second rate the worm generates.

For example, if the total number of vulnerable systems is 10, the number of possible addresses is 100, and the initially infected systems equals 1, then the Matlab code would generate one random number for the first iteration. If the random number generated falls within the range of 1-10 then the iteration number is noted by the code and the number of vulnerable systems is reduced to 9. On the next iteration, the Matlab code generates one random number for each of the two infected systems and compares those numbers to the remaining vulnerable systems. An example of 5 iterations of the Matlab model code, including the first two iterations detailed above, are demonstrated in Table 5 for a worm packet per second rate of 1 second.

Table 5. Matlab Model Infection Rate Example

Iteration	Number of Infected Systems	Number of Vulnerable Systems	Number of Addresses Available	Number of Pseudo Random Number Generated	Numbers Generated	Time of Infection in Seconds
1	1	10	100	1	7	1
2	2	9	100	2	83 & 59	2
3	2	9	100	2	3 & 60	-
4	3	8	100	3	12, 33, 72	4
5	3	8	100	3	45, 51, 90	-

From this information a graph of the infection rate curve can be generated as shown in Figure 11 where the data in Table 5 is extended exponentially.

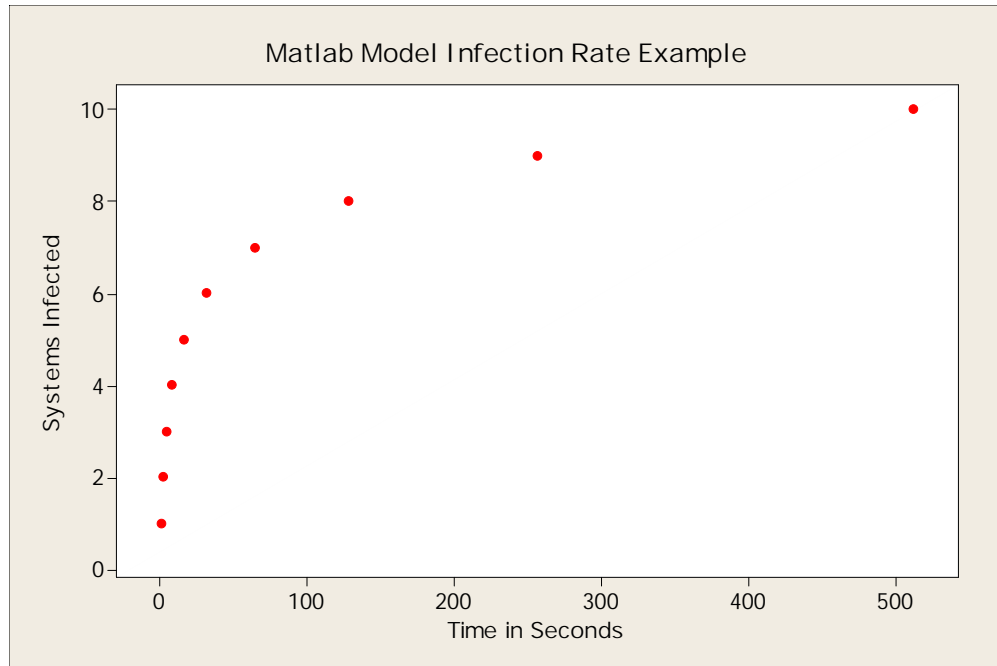


Figure 11. Matlab Model Infection Rate Example

#### 3.10.4 Experimental Validation of Matlab Model Infection Rate Simulation

Once the uniform distribution of the random packets generated by a Slammer-infected system is verified, the Matlab model of worm infection rates is validated. The verification of the mathematically generated infection rate is accomplished by comparing the curve generated by the Matlab model code (Section 3.10.3 and Appendix C) against the curves generated by previous research and the original observations of the Slammer worm from 2003. The Matlab model is first compared against the Code Red worm, the Code Red “/8” routing worm, and the Code Red BGP routing worm experiments [ZTG05]. A more detailed comparison is performed against the Slammer worm’s original infection rate and the results found in the research of the Slammer worm completed by Wei and Zou [WMS05] [ZTG05]. Finally, the proposed SSE routing worm is compared against the Matlab models of the Slammer worms of 2003/2007 and the Slammer routing worms of 2003/2007.

As discussed in Section 3.10.3, the Matlab model generates an infection rate curve based on the variables of the initially infected systems, the number of vulnerable systems, and the size of the address space being scanned by the worm. The Matlab model variables used to generate the infection rate curves for each of the worms tested in this research are described below and in Appendix C.

#### 3.10.4.1 Matlab Model Variables for the Code Red Worm Comparisons

The Code Red worm simulations have a vulnerable population of 360,000 and an initial number of infected systems of 10. The scan rate for the Code Red test is based on the observed rate of 358 scans per minute. These values and the 4.3 billion possible addresses are provided to the Matlab model simulation as the input variables. For ease of reference, Table 6 provides all of the variables used in the Matlab model for the three Code Red simulations.

The Zou Code Red “/8” routing worm simulations consist of a vulnerable population of 360,000 and an initial number of infected systems of 10. The scan rate for the Zou Code Red “/8” routing worm test remains at the observed Code Red rate of 358 scans per minute. However, to mimic the operation of the “/8” routing worm’s ability to reduce the address space required to scan the possible addresses are reduced to 1,946,156,941.

The Zou Code Red BGP routing worm simulations have of a vulnerable population of 360,000 and an initial number of infected systems of 10. The scan rate for the Zou Code Red BGP routing worm test remains at the observed Code Red rate of 358 scans per minute. However, because the Zou Code Red BGP routing worm further

refines the address space required for scanning the possible addresses are reduced to a total of 1,228,360,647.

Table 6. Matlab Model Variables for Code Red Worms

	Number of IP addresses	Number of Vulnerable Systems	Number of Iterations	Number of Trials	Initial Number of Infected Systems
Code Red Worm	4,294,967,296	360,000	10,000,000	1	Ten
Zou Code Red "/8" Routing Worm	1,946,156,941	360,000	500,000	1	Ten
Zou Code Red BGP Worm	1,228,360,647	360,000	500,000	1	Ten

Due to the number of vulnerable systems, and the results of the trial tests, this experiment is run only once as an evaluation of the Matlab model infection rate. The remaining worm tests are examined more thoroughly to establish the validity of the Matlab model.

#### 3.10.4.2 Matlab Model Variables for the Zou Slammer Worm Comparisons

At the time of Slammers release in January of 2003 the vulnerable population was estimated at 75,000 with 74,856 unique IP addresses observed [MPW03]. Unfortunately, the numbers used by Zou was arbitrarily set to 100,000 for the number of vulnerable systems and 10 for number of initially infected systems [ZTG05]. The scan rate used for the Zou Slammer worm is 4,000 packets per second, which simulates the observed average of Slammer worm during its original run. Table 7 provides the Matlab model variables used in the Zou Slammer worm simulations.

Table 7. Matlab Model Variables for Zou Slammer Worms

	Number of IP addresses	Number of Vulnerable Systems	Number of Iterations	Number of Trials	Initial Number of Infected Systems
Zou Slammer Worm	4,294,967,296	100,000	10,000,000	20	Ten
Zou Slammer Routing Worm	1,946,156,941	100,000	500,000	20	Ten

The Zou Slammer routing worm is run in the Matlab model with the same 10 initially infected systems as the Zou Slammer worm. However, the number of IP addresses and scanning rate are changed to reflect the capabilities of the Zou Slammer worm. The number of IP addresses available for scanning by the Zou Slammer routing worm is reduced to 1,946,156,941 to reflect the worm's ability to reduce the required scanning space and the scan rate is decreased to 3,108 pps to reflect the larger packet size of 520 bytes.

The Slammer worm simulations are run in the Matlab model 20 times and used to generate a mean infection rate curve with confidence level of 95%.

#### 3.10.4.3 Matlab Model Variables for the Slammer Worm Comparisons

To more accurately represent the actual operation of the original Slammer worm, number of vulnerable systems is set to the 74,856 unique IP addresses observed in 2003 is used for the Slammer worm 2003 model. Additionally, the number of initially infected systems is set to one since there was no evidence to support there was originally more than one initially infected system. The scan rate used for the Slammer worm 2003 is set at the originally observed 4,000 packets per second. Table 8 provides the Matlab model variables used in the Slammer worm 2003 and 2007 simulations.

The Slammer worm 2007 is run using the Matlab model with the same number of initially infected systems and vulnerable systems as the Slammer worm 2003. However, the scanning rate is changed to 14,398 pps to reflect the observed capabilities of computing systems today.

The Slammer worm simulations are run in the Matlab model 20 times and used to generate a mean infection rate curve with confidence level of 95%.

Table 8. Matlab Model Variables for the Slammer Worms

	Number of IP addresses	Number of Vulnerable Systems	Number of Iterations	Number of Trials	Initial Number of Infected Systems
Slammer Worm 2003	4,294,967,296	74,856	10,000,000	20	One
Slammer Worm 2007	4,294,967,296	74,856	10,000,000	20	One

#### 3.10.4.3 Matlab Model Variables for the Slammer Routing Worm Comparisons

Due to the use of an arbitrary number of 100,000 vulnerable systems and 10 initially infected systems by Zou, the infection rate curve they generated does not accurately represent how a Slammer routing worm would have behaved in 2003. To correct this problem, the Matlab model is used to generate a Slammer routing worm using the original 2003 Slammer worm numbers. Thus, the number of vulnerable systems is set to the 74,856 unique IP addresses and the number of initially infected systems is set at one. The scan rate used for the Slammer routing worm 2003 is set at the originally observed 3,108 packets per second as calculated by Zou because of the increase of the infection packet size to 520 bytes [ZTG05]. Table 9 provides the Matlab model variables used in the Slammer routing worm 2003 and 2007 simulations.

The Slammer routing worm 2007 is run using the Matlab model with the same number of initially infected systems and vulnerable systems as the Slammer routing worm 2003. However, the scanning rate is changed to 11,187 pps to reflect the increased packet generation capabilities of computing systems today.

The Slammer routing worm simulations are run in the Matlab model 20 times and used to generate a mean infection rate curve with confidence level of 95%.

Table 9. Matlab Model Variables for the Slammer Routing Worms

	Number of IP addresses	Number of Vulnerable Systems	Number of Iterations	Number of Trials	Initial Number of Infected Systems
Slammer Routing Worm 2003	1,946,156,941	74,856	500,000	20	One
Slammer Routing Worm 2007	1,946,156,941	74,856	500,000	20	One

#### 3.10.4.4 Matlab Model Variables for the SSE Routing Worm Comparisons

For the analysis of the SSE routing worms proposed in this research, the number of vulnerable systems is reduced to the size of one “/8” grouping of IP addresses. The total 74,856 vulnerable systems are divided by the 116 “/8” routable addresses. This makes the total vulnerable population within each “/8” set scanned by the SSE routing worm 645 systems. The number of IP addresses scanned by the SSE routing worm is also reduced to 16.7 million. This represents the total of the 1.95 billion available for normal Slammer routing worm divided into the 116 “/8” ranges. Due to the size of the SSE routing worm only increasing the original Slammer worm code by eight bytes, the packets per second rate is set at 3,922. Table 10 provides the Matlab model variables used in the SSE routing worm 2003 and 2007 simulations.

The SSE routing worm 2007 is run in the Matlab model with the same number of initially infected systems and vulnerable systems as the SSE routing worm 2003.

However, the scanning rate is changed to 14,118 pps to reflect the increased packet generation capabilities of computing systems today.

The SSE routing worm simulation is run in the Matlab model 50 times using the previously identified 16.7 million addresses and 645 vulnerable systems. This curve is compared to the previous worm simulations for analysis of similarity.

Table 10. Matlab Model Variables for the SSE Routing Worms

	Number of IP addresses	Number of Vulnerable Systems	Number of Iterations	Number of Trials	Initial Number of Infected Systems
SSE Routing Worm 2003	16,777,216	645	500,000	50	One
SSE Routing Worm 2007	16,777,216	645	500,000	50	One

### 3.10.5 Experimental Examination of the Slammer Scanning Rate

Packet generation rate by original Slammer worm is collected from the testbed network to provide another point of comparison with the real world observations. The packets are collected after the initial infection is complete and the infected system is automatically generating infection packets. The mean difference in inter-arrival time between each packet is measured for the collection period. Twenty separate collections of infection packets are used to generate the packet per second rate.

From the packet generation analysis, the speed of an average 2007 computing system is estimated and used to generate a comparison between the worm speeds of today versus those possible in 2003. For these tests, the original Slammer worms, the Slammer



routing worms, and the SSE routing worms are set to their respective available address scanning spaces and vulnerable systems as shown in Table 11.

Table 11. Worm Variables for Year Simulations

	Number of IP addresses	Number of Vulnerable Systems	Number of Iterations	Number of Trials	Initial Number of Infected Systems
Slammer Worm 2003	4,294,967,296	74,856	10,000,000	20	One
Slammer Worm 2007	4,294,967,296	74,856	10,000,000	20	One
Slammer Routing Worm 2003	1,946,156,941	74,856	500,000	20	One
Slammer Routing Worm 2007	1,946,156,941	74,856	500,000	20	One
SSE Routing Worm 2003	16,777,216	645	500,000	50	One
SSE Routing Worm 2007	16,777,216	645	500,000	50	One

### 3.10.6 Examination of the Slammer Infection Doubling Rate

From the Matlab model simulations of the Slammer worm 2003 variable numbers an infection doubling rate is developed for comparison to the 8.5 (+/-1) second estimated by Moore [MPW03]. This estimated global infection doubling rate is calculated for 65,536 systems, which is the largest number of vulnerable systems attainable prior to the limit of 74,856 possible systems. The Moore research noted, the original Slammer worm doubling rate was only estimated for the first minute [MPW03]. Therefore, in addition to the complete doubling rate curve, a more detailed analysis of the first 60 seconds is provided. Then Matlab model of the Slammer worm 2003 doubling rate is compared original Slammer worm infection doubling rate curve for evaluation of their similarity.

### **3.11 Analysis and Interpretation of Results**

There are many aspects of the Slammer worm investigated in this research, the first of which is the packet per second generation speed. The packets generated per second by the Slammer worm infected system on the testbed network are analyzed and compared to the observed number of packets per second generated by Slammer in January 2003. This provides a basis for the calculating the time for each potential victim to be infected on the current Internet, which allows the speed of infection to be determined that could be expected if the worms were released on a network today.

This research investigates the Slammer worm's ability to generate a uniform random IP addresses in a given address space through the use of statistical comparison. Based on the comparison of the destination IP addresses of infection packets collected to the statistically generated regression line the uniform distribution of the Slammer worm's IP address generation is established. The similarity between these is evaluated to establish the uniformity of distribution. Further, to determine if the Slammer worm is generating the random IP addresses in a non-random pattern, lag plots of the data are created to analyze for existence of an observable pattern. The presence of a pattern demonstrates the numbers are in fact correlated. The combination of the analysis of the uniform distribution and lag plots provide this research to determine the statistical randomness of the Slammer worm's generation of IP addresses.

The Matlab model of worm infection rates is compared to previous research for validation and analysis of the original Slammer worm, the Slammer routing worm, and the SSE routing worm. The first Matlab Model comparison made is to the Code Red worm research performed by Zou [ZTG05]. The Matlab model of the Code Red worm,

the Zou Code Red BGP routing worm, and the Zou Code Red “/8” Routing is analyzed for similarity to the of the Zou infection rates of those worms [ZTG05]. The Matlab model of the original Slammer worm infection rate is compared to the observed Slammer characteristics from January 2003 and to the results of the Wei and Zou research [WMS05], [MPS03], [MPW03], and [ZTG05]. As part of this comparison of the Matlab model to the Slammer worm’s operational characteristics, the Matlab model rate of infected systems doubling versus that observed by Moore is performed [MPW03]. If the comparisons show a close relationship, then the Matlab model simulations can be considered representative of how the original Slammer worm operates.

The Matlab model is used to generate the infection rate curves for comparison of the original Slammer worm 2003 and how the Slammer worm operates on a system of today. The Slammer routing worm is also generated by the Matlab model for the 2003 and 2007 computing system operation. These four worms, the Slammer worm 2003/2007 and the Slammer routing worm 2003/2007, are compared to determine the speed differences between each other. These differences will determine how fast each worm is in relation to each other and characterizes their infection rate.

Finally, the SSE routing worm is modeled by Matlab to characterize its infection rate. Then the SSE routing worm 2003/2007 are calculated and compared to the Slammer worm 2003/2007 and Slammer routing worm 2003/2007 for determination of the speed difference between them. These final comparisons provide the ability to determine which of these worms is the fastest at infecting a vulnerable population.

### **3.12 Summary**

This chapter discusses the methods used to analyze the propagation of the original Slammer worm, the Slammer Routing worm, and the SSE routing worm across an IPv4 network. This chapter defines under consideration for this research, the goals and hypothesis, the approach taken to complete this experiment, and the assumptions and limitations bounding this research. In this chapter, the system boundaries, the services provided by the system, and the workload of the system is covered. The metrics used in the measurement of performance, the description of the system and workload parameters, and the factors are discussed in this chapter. The techniques used for evaluating the data and a detailed explanation of the experiment design is provided. This chapter concludes with coverage of the analysis and interpretation of the results generated by the experiments in this research.

## IV. Analysis and Results

This chapter presents the results and analysis of the data collected from the experiment simulations of the worm infections. Section 4.1 covers the collection and analysis of Slammer's packet per second generation. Section 4.2 examines the randomness of Slammer's IP address and octet generation. In Section 4.3 the Matlab worm models are presented for comparison to the original Slammer worm and the routing worm models proposed by Zou. The comparison of worm speeds possible on computing systems of today versus those available in 2003 is provided in Section 4.4. Section 4.5 examines the infection rate of the original Slammer worm versus the SSE routing worm.

### 4.1 Slammer Packet Generation

The test run to identify a mean packet generation time by an infected system yielded interesting results. Twenty separate collections of packets generated by the original Slammer worm code are collected and analyzed. As shown in Table 12, the mean time between packet generation is 69.46  $\mu$ sec with the laptop operating on A/C power. This generation time is extremely stable, as seen by the small standard deviation and confidence intervals. This equates to 14,398 packets per second (pps), which falls into the upper half of the range observed during the Slammer worm's original release of an average 4,000 and maximum 26,000 scans per second per worm-infected machine [MPS03].

Table 12. Slammer Packet Per Second on A/C Power

Average Inter-Arrival Time of Infection Packets	Standard Deviation	Upper Confidence Interval @95%	Lower Confidence Interval @ 95%	Packets Per Second or Scans Per Second
69.46 $\mu$ sec	1.49 $\mu$ sec	70.11 $\mu$ sec	68.8 $\mu$ sec	14398 pps

The testing of Slammer's packet per second capability reveals an increase in the time to generate packets when the system is operating on battery power. A short test and analysis of this anomaly is performed to calculate the packets per second of Slammer on battery power. As shown in Table 13, the infected system generates less than half of the amount of packets on battery power as it does on A/C power. The mean time between packet generation is 152.28  $\mu$ sec with the laptop operating on battery power. The standard deviation and confidence intervals are only slightly larger than those observed while operating on A/C showing that the system is still fairly stable in the packet per second generation while operating on battery power. This generation rate equates 6,567 pps on battery power. Although this amount is less than half of the observed amount when operating on A/C power, Slammer is still generating over the average 4,000 and below the maximum 26,000 scans per second per worm-infected machine observed on the Internet [MPS03].

Table 13. Slammer Packet Per Second on Battery Power

Average Inter-Arrival Time of Infection Packets	Standard Deviation	Upper Confidence Interval @95%	Lower Confidence Interval @ 95%	Packets Per Second or Scans Per Second
152.28 $\mu$ sec	6.01 $\mu$ sec	155.55 $\mu$ sec	149.02 $\mu$ sec	6,567 pps

This data confirms that the Slammer code being utilized is operating within the previously observed characteristics and validates the use of the A/C power packets per second for use in calculating the time it will take to infect a group of systems of similar

construction. The validation of the Matlab model infection rates will initially be validated using 4,000 scans per second per worm [ZTG05], [WMS05], [MPW03].

Moore's Law has been refined and changed over the years. Adjustments had to be made due to observations where Moore's Law fell short by 38% during a 1970's estimate and was over by 27% in 1975 of transistor capabilities. The original version of the law developed in 1965, was that the doubling would occur every 12 months. Later, there was some consideration given to increase the time span to 24 months. This refinement led to the currently quoted estimation of 18 months.

Due to this ever-changing growth rate, this 18-month rule has been refined further to try and match the observed growth curves. Dave Epstein of the *Microprocessor Report* suggested a solution to the variation in Moore's Law called "Epstein's amendment." Epstein's amendment modifies the growth calculations of Moore's Law by adding an additional six months to the doubling rate on an every ten years interval. Where in the 1970s the growth rate was every twelve months, by 1980 the doubling rate needed to be increased to every 18 months. Finally, by Epstein's amendment, the growth rate in 2000 was projected to double every 30 months [Hal06].

In consideration of Moore's Law and Epstein's modification, the operational characteristics of systems currently connected to the Internet should approach the average Slammer packet per second generation of 3.2 times that originally observed in January of 2003. Thus, a network of computers in January 2007 should produce an average of 12,800 pps when infected by Slammer. The observed packet per second rate for a Slammer infected system during this research is 14,398 pps. The observed packet per second generation is 12.48% greater than that of the Moore's Law with the Epstein

modification estimate. However, the estimated 2007 upper limit of scan per second is calculated at 83,200 using Moore's Law with the Epstein amendment. This makes the 14,398 pps observed during this research fall within the estimated range from the average of 12,800 to a maximum of 83,200 for a computing system of today. Thus, for final speed analysis of a computing system in 2007 the average observed results for Slammer packet generation on an A/C power source of 14,398 pps is used.

## **4.2 Slammer Randomness**

The true randomness of the original Slammer worm's pseudo random number generator has previously been studied by Moore [MPS03]. In that analysis they observed a small flaw that limited the ability of the original Slammer worm to generate random numbers, the corrections of which are discussed in more detail in Appendix B [MPS03]. However, this flaw did not prevent Slammer from essentially taking over the Internet during its run [HyE03]. In these next two subsections the analysis of Slammer's capability to uniformly generate random IP Addresses across the IP address range under test is presented. First, Slammer's ability to generate a uniform distribution of IP's across the entire address span under test is considered. Then Slammer's ability to generate a uniform random distribution across the individual last three IP address octets is examined.

### **4.2.1 Slammer IP Address Randomness**

The analysis provided here shows that despite its flaws, the original Slammer worm generated IP addresses are well distributed throughout the address space. However, as a whole there is a pattern in the form that the addresses are generated. This could be indicative of the flaw noted by Moore [MPS03] [MPW03].



The lag plot in Figure 12 of the IP addresses generated by the Slammer worm is a typical example of 100,000 data points obtained during testing. This lag plot of the combined octets generated by Slammer shows a pattern in the IP address generation. The addresses that are generated seem to develop a repeating diamond pattern formed by two faintly intersecting lines. From this lag plot it is clear that there is an observable pattern to the generation of random numbers by Slammer. Despite this anomaly, the worm provides a fairly uniform distribution for the statistical coverage of the entire range of IP addresses being considered in this experiment.

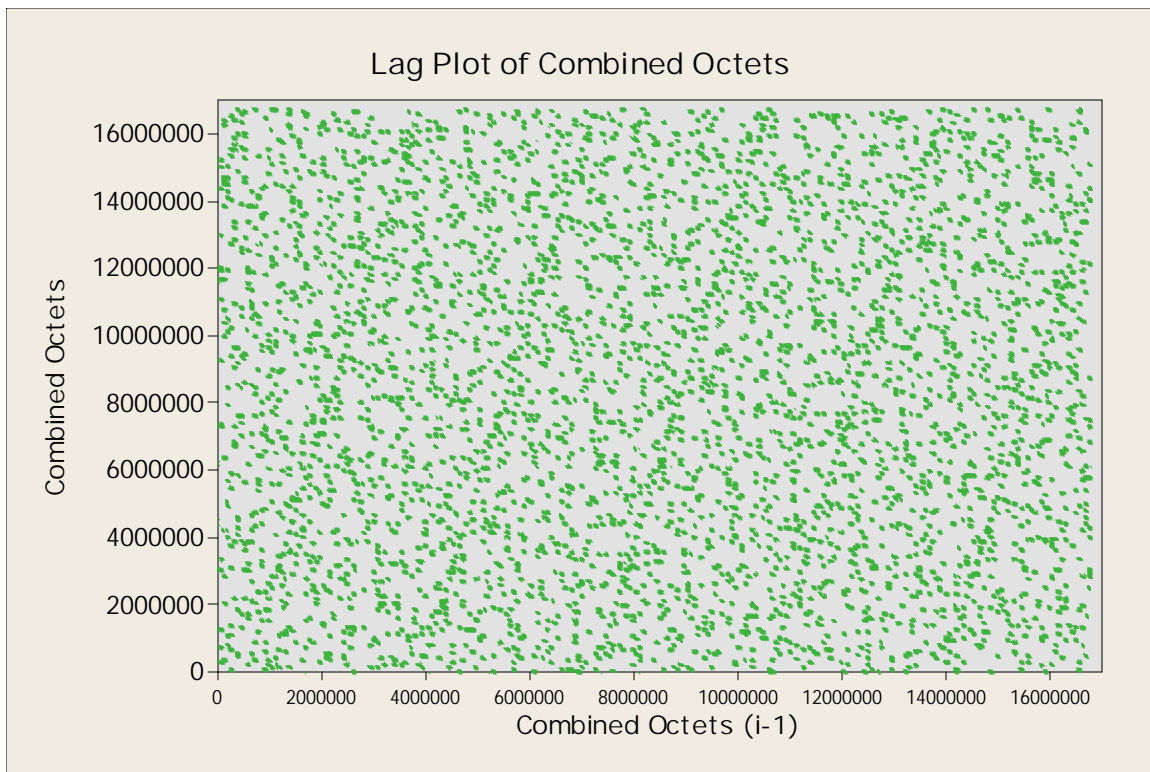


Figure 12. Lag Plot of Slammer-Generated IP Addresses

The regression equation of the Slammer IP addresses gathered from the infection packets, as shown in Table 14, shows that the p-value for the comparison to the

statistically generated IP address line is zero. This means there is a high correlation between the statistical and Slammer-generated IP addresses. Further, the R-squared values, which represent the closeness of fit to a linear line, are both 100% indicating that the Slammer-generated packets match the statistical line

Table 14. Slammer IP Address Generation Regression Analysis

The regression equation is					
Slammer Generated = -197.4 + 1.00 Statistically Generated					
S = 1945.04		R-Sq = 100%		R-Sq(adj) = 100%	
Analysis of Variance					
Source	DF	SS	MS	F	P
Regression	1	1.53721E+18	1.53721E+18	4.06328E+11	0.000
Error	65533	2.47923E+11	3.78318E+06		
Total	65534	1.53721E+18			

The residual plots, shown in Figure 13, further demonstrate the even distribution of the Slammer-generated IP addresses. Although there is a large variation in the calculated value of residuals, this can be explained by the size of the address space compared to the number of samples. Because the address space under consideration is 4.3 billion possible addresses and the number of samples used in analysis limited by software restrictions and memory limits, the residual analysis is based on 0.0015 percent of the possible addresses available. This numerical limitation is causing the larger variation in residuals shown. However, both visually and mathematically, the plots show that the differences between the IP addresses generated by Slammer are uniform across the range of the address space. The normal probability plot illustrates that the generated packets match the statistical line with only a small deviation at the tails of less than one percent. Both the fitted value and observation order of the residuals show that there is an even distribution of differences across the IP address range. Finally, the histogram of the

residuals shows a well centered distribution with tails that fall away at a smooth rate. The histogram strongly indicates the Slammer-generated IP addresses and the statistically generated IP addresses are the same.

The fitted line plot in Figure 14 demonstrates both visually and mathematically that the IP addresses generated by Slammer and the statistical model are identical. As shown previously in the analysis of the residuals, this fitted line plot analysis calculates the R-squared values to be a 100% match.

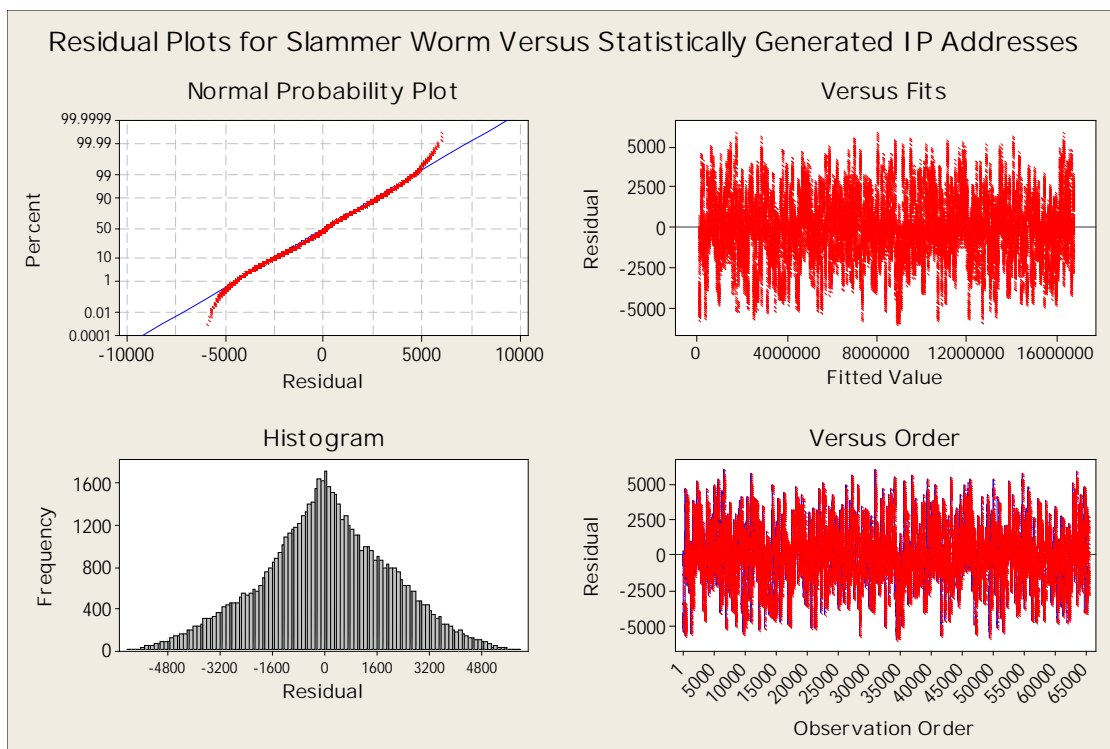


Figure 13. Residual Plots of Slammer IP Address Generation

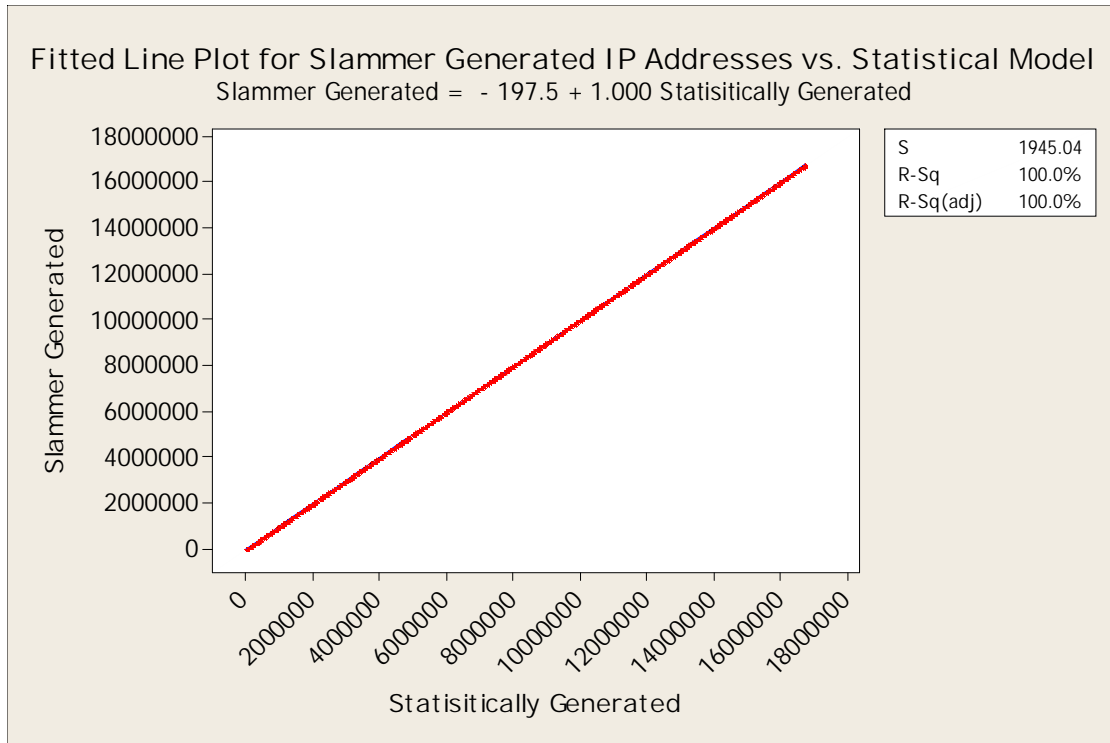


Figure 14. Fitted Line Plot of Slammer versus Statistical Model

Thus, despite the Slammer-generated IP address pattern observed in Figure 12, the IP addresses generated are shown to be uniformly distributed across the IP range being considered. This uniform distribution ensures the experiment will accurately represent the simulation of a randomly scanning worm across the IP address space.

#### 4.2.2 Slammer Octet Randomness

In this subsection each individual octet generated by Slammer is examined to determine the uniformity of distribution throughout the octet range. This uniform distribution demonstrates a statistical coverage, in that the entire range is of addresses is chosen with equal probability, over the entire range of octets being considered in this experiment. The detail provided in this section further validates the ability of a Slammer-based worm to randomly generate any octet value for a given IP range. Each octet is individually examined to provide fine granularity for emphasis on the uniform

distribution within the set of randomly generated octets. Each individual lag plot is a typical example of a collection of 2,560 data points. This provides the clearest visual indication of whether a trend in the data is present. However, the regression equations and residual plots are attained through the use of the same typical 65,534 data point captures.

#### 4.2.2.1 Slammer Second Octet Randomness

Shown in Figure 15, the lag plot of the second octet demonstrates some of the previously observed anomalies that were found during the evaluation of the combined octet generation lag plot. Here the lines are harder to discern, but the lines are still visually observable illustrating that there is some pattern to the generation of the octets.

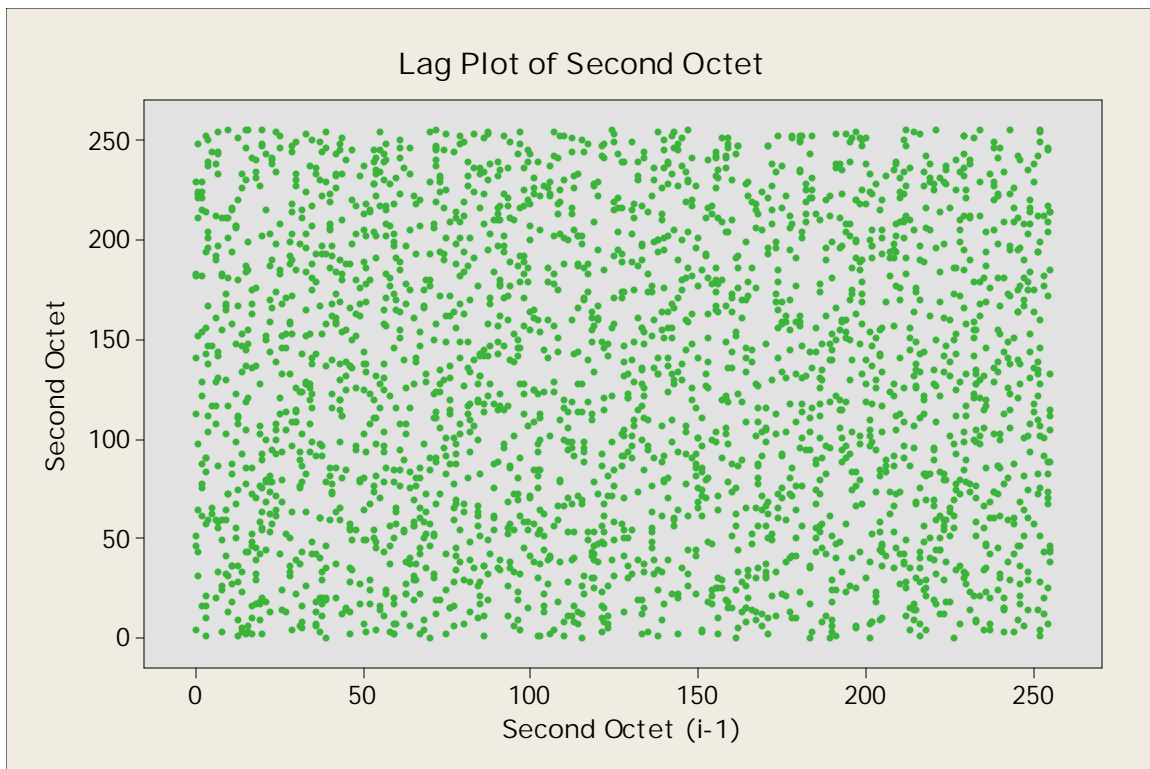


Figure 15. Lag Plot of Slammer-Generated Second Octet

However, once again the regression equation of the Slammer generation of the second octet, as shown in Table 15, shows a value of zero for the p-value for the comparison to the statistically generated octet series. This means there is a high correlation between the statistical and Slammer-generated random octet series. Further, the R-squared values, which represent the closeness of fit to a linear line, are both 100% indicating that the Slammer-generated packets match the statistical line.

Table 15. Slammer-Generated Second Octet Regression Analysis

The regression equation is					
Slammer Generated = -0.5422 + 0.9998 Statistically Generated					
S = 0.300637		R-Sq = 100%		R-Sq(adj) = 100%	
Analysis of Variance					
Source	DF	SS	MS	F	P
Regression	1	357748656	357748656	3.95816E+09	0.000
Error	65533	5923	0		
Total	65534	357754579			

The residual plots, shown in Figure 16, further demonstrate the even distribution of the Slammer-generated IP addresses. The 4-way plot shows that the differences between the octets generated by Slammer and the statistical model are uniform across the range of the address space. The normal probability plot illustrates that the generated packets match the statistical line with some small deviation at the tails of less than five percent. Both the fitted value and observation order of the residuals show that there is an even distribution of differences across the octet range, but visually there appears to be some possible sinusoidal pattern to the octets generated. This sinusoidal modulation is small with a +/- 0.25 difference in the residuals. The histogram of the residuals illustrates a visually unusual pattern with the distribution of the histogram being normally distributed with a flattened top. This flattened top is not due to graphical clipping, but indicates that the differences in the residuals are evenly spread across the octet address

span. Despite these unusual visual patterns, the statistical analysis of the regression equation denotes the data is statistically uniformly distributed across the address space.

In the fitted line plot in Figure 17, it is shown that both visually and mathematically that the second octet generated by Slammer and the statistical model are identical. As shown previously in the analysis of the residual equation and plots, the fitted line plot analysis shows the R-squared values to be a 100% match.

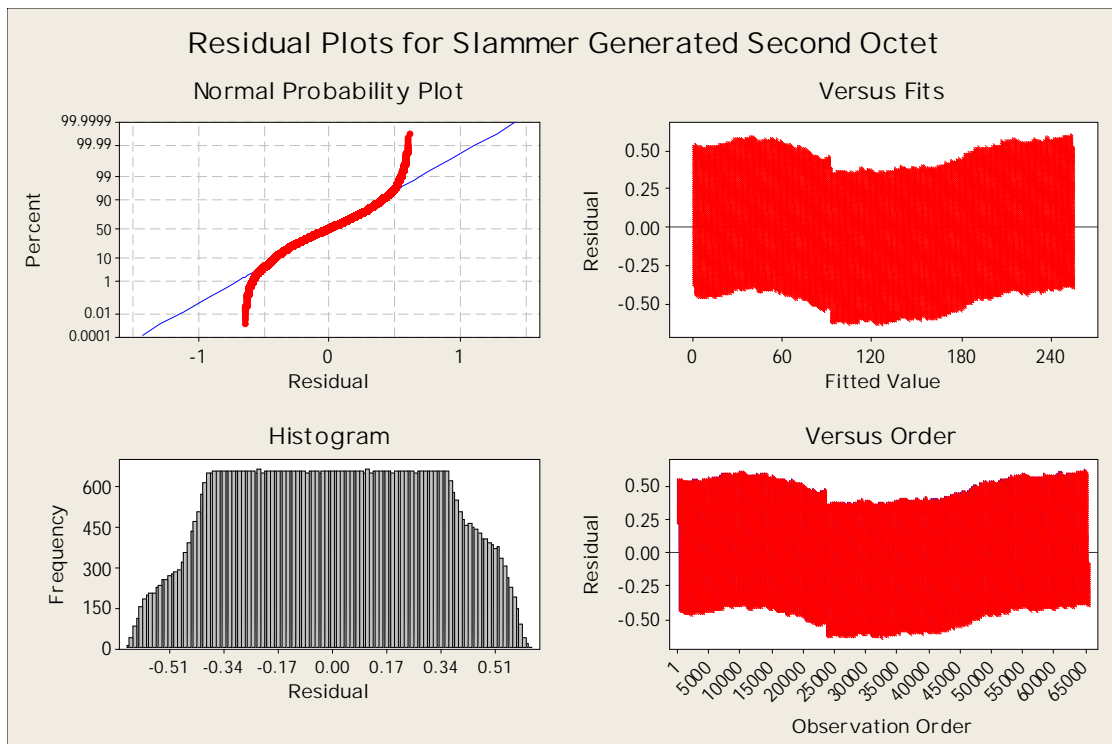


Figure 16. Residual Plot for Slammer-Generated Second Octet

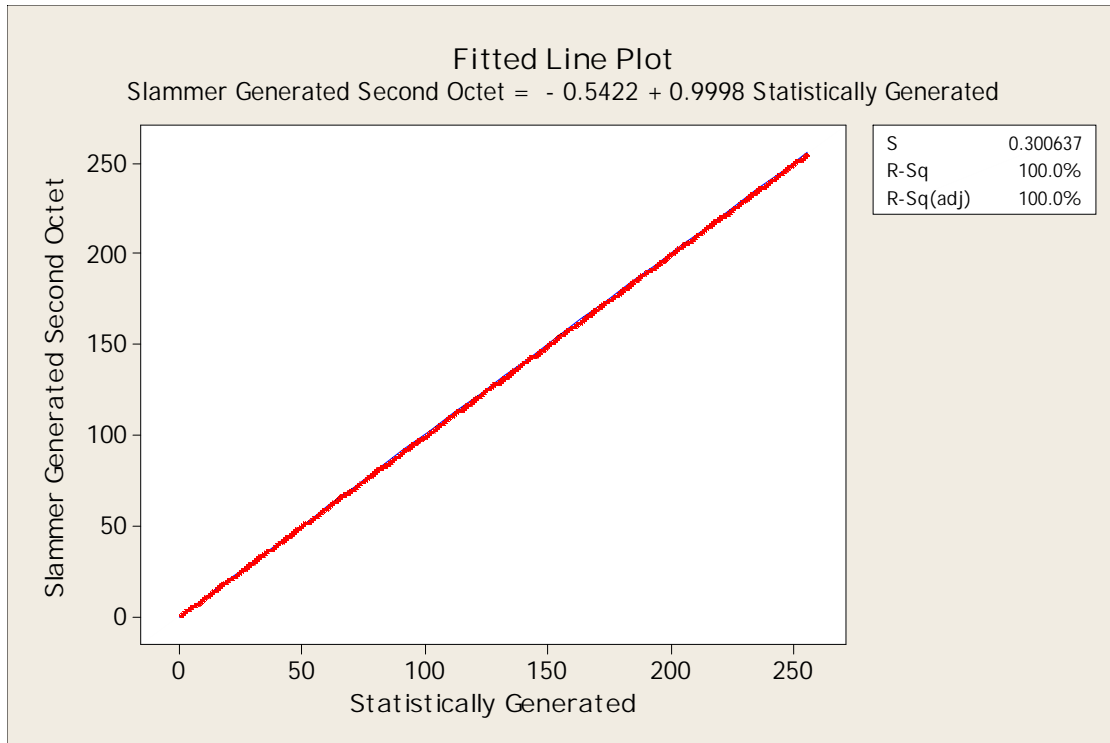


Figure 17. Fitted Line Plot for Slammer-Generated Second Octet

#### 4.2.2.2 Slammer Third and Fourth Octet Randomness

The analysis for the third and fourth octets generated by Slammer reveals results that are almost identical. Unlike the combined octet and second octet lag plots, the lag plots for the third and fourth octets shown in Figures 18 and 19 do not exhibit any discernable address generation pattern. This provides further evidence that the pattern observed in the combined and second octet lag plot analyses are related and not necessarily pervasive throughout the original Slammer worm random address generation code. This also indicates that the diamond pattern observed is related to the errors denoted in the Slammer worm analysis performed by Moore [MPS03]. Further detailed analysis to pinpoint this anomaly in the random number generation algorithm anomaly is not part of this research; however this may need to be investigated prior to extending the experiment presented here.



Another indication in the similarity between the third and fourth octets is the extremely small difference in their regression equations as shown in Tables 16 and 17. Due to this similarity, their analysis is combined for brevity. The regression equations for the Slammer generation of these last two octets each have a zero for their p-value. This proves there is a high correlation between the statistical and Slammer-generated random octets. The R-squared values for each of the last two octets equal 100%. This analysis shows, once again, that the random octets generated by Slammer match the statistical line.

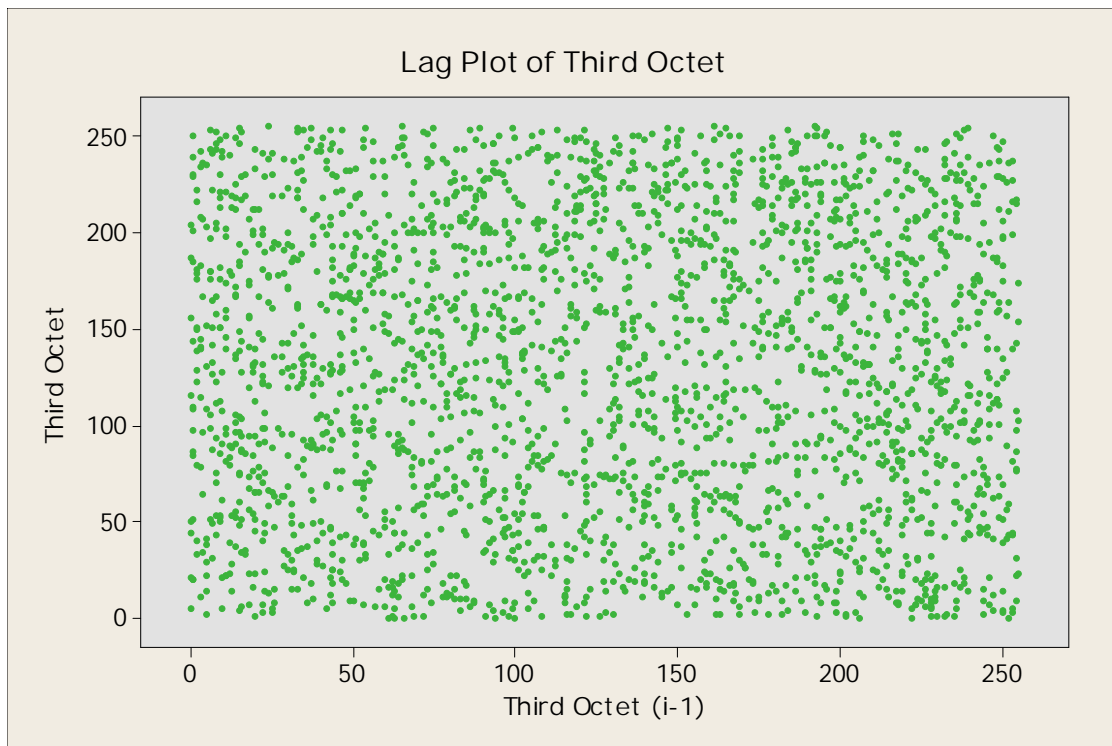


Figure 18. Lag Plot of Slammer-Generated Third Octet

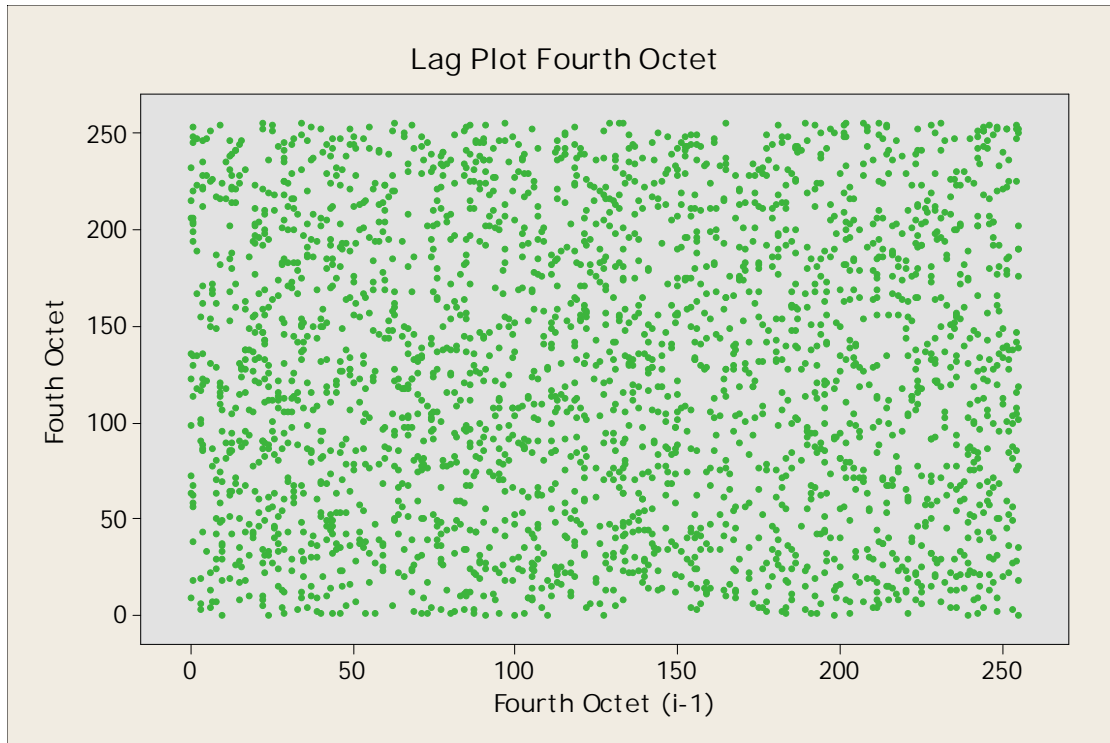


Figure 19. Lag Plot of Slammer-Generated Fourth Octet

Table 16. Slammer-Generated Third Octet Regression Analysis

The regression equation is					
Slammer Generated = -0.5076 + 0.9996 Statistically Generated					
S = 0.322242		R-Sq = 100%		R-Sq(adj) = 100%	
Analysis of Variance					
Source	DF	SS	MS	F	P
Regression	1	3576561334	3576561334	3.44431E+09	0.000
Error	65533	6805	0		
Total	65534	357662938			

Table 17. Slammer-Generated Fourth Octet Regression Analysis

The regression equation is					
Slammer Generated = -0.517399 + 0.999 Statistically Generated					
S = 0.332113		R-Sq = 100%		R-Sq(adj) = 100%	
Analysis of Variance					
Source	DF	SS	MS	F	P
Regression	1	357164896	357164896	3.23815E+09	0.000
Error	65533	7228	0		
Total	65534	357172124			

The residual plots for the third and fourth octets, shown in Figures 20 and 21, continue to illustrate the even uniform distribution of Slammer infection packets. The 4-way residual plots display that the differences between the octets generated by Slammer and the statistical model are uniform across the range of the octet address space. The normal probability plot illustrates that the generated packets match the statistical line with some small deviation at the tails approaching one percent. Both the fitted value and observation order of the residuals show that there is an even distribution of differences across the octet range, but as observed in the analysis of the second octet, visually there appears to be some sinusoidal pattern to the octets generated. This sinusoidal modulation is double the characteristics observed with the second octet with a  $\pm 0.50$  difference in the residuals. However, unlike the histogram of the second octet, the histogram of the residuals for the third and fourth octets illustrate a more normally distributed pattern with only a small indication of a flattened top. Although there appears to be a more pronounced sinusoidal pattern in the fitted and observed residuals, the statistical analysis of the regression data indicates that the data is statistically uniformly distributed.

The fitted line plots shown in Figures 22 and 23 demonstrate both visually and mathematically that the last two octets generated by Slammer and the statistical model are virtually identical. As shown previously in the analysis of the residual equation and plots, the fitted line plot analysis shows the R-squared values to be a 100% match.

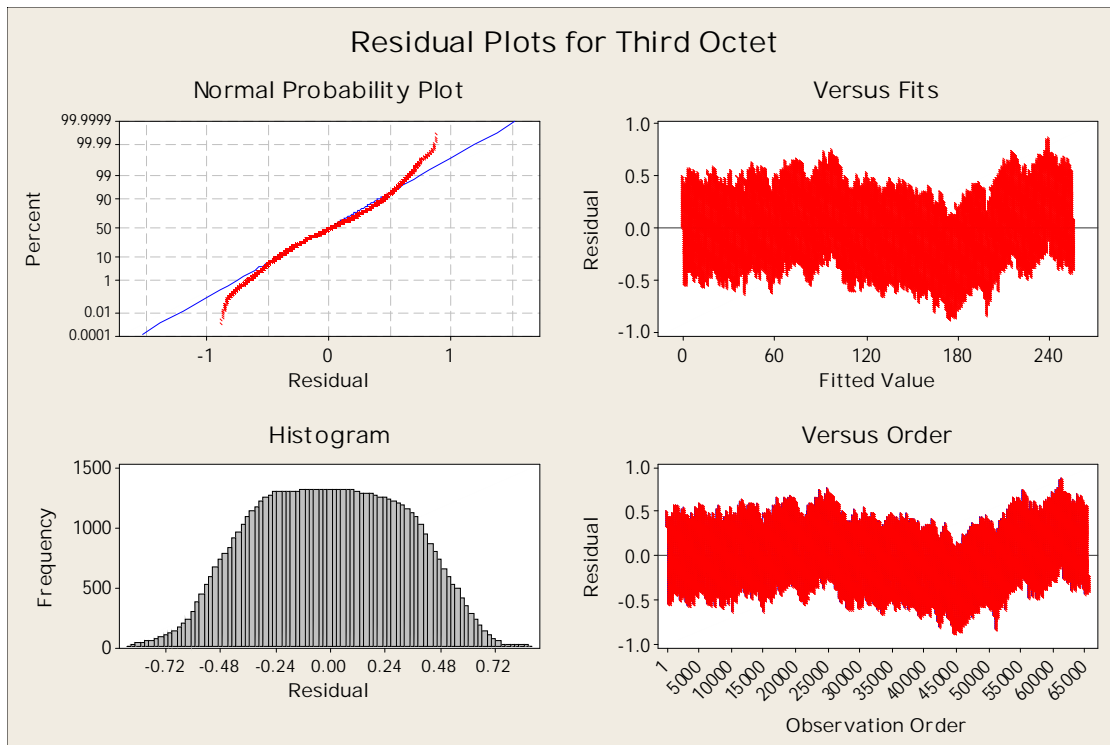


Figure 20. Residual Plot for Slammer-Generated Third Octet

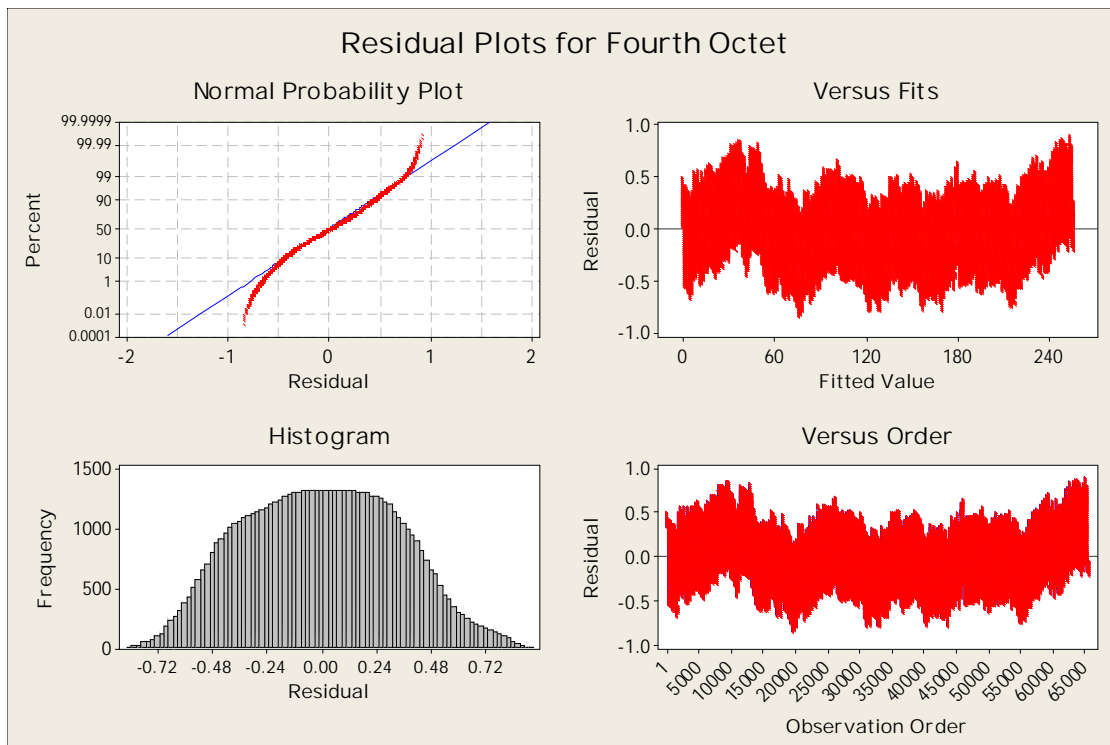


Figure 21. Residual Plot for Slammer-Generated Fourth Octet

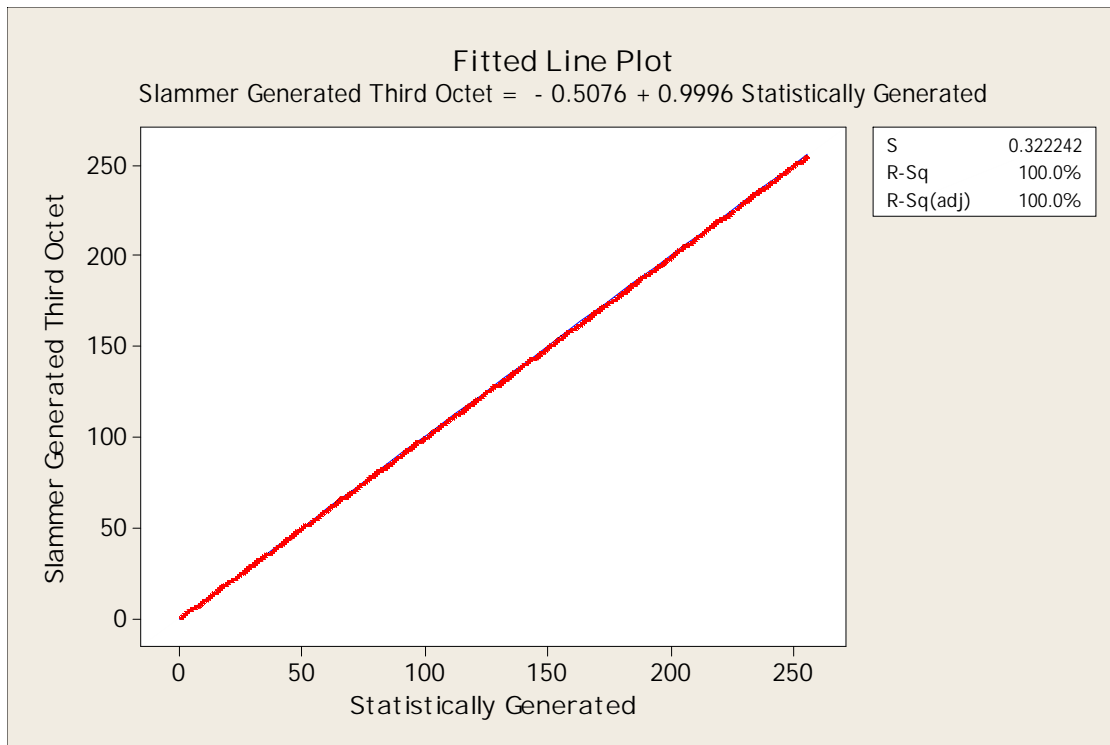


Figure 22. Fitted Line Plot for Slammer-Generated Third Octet

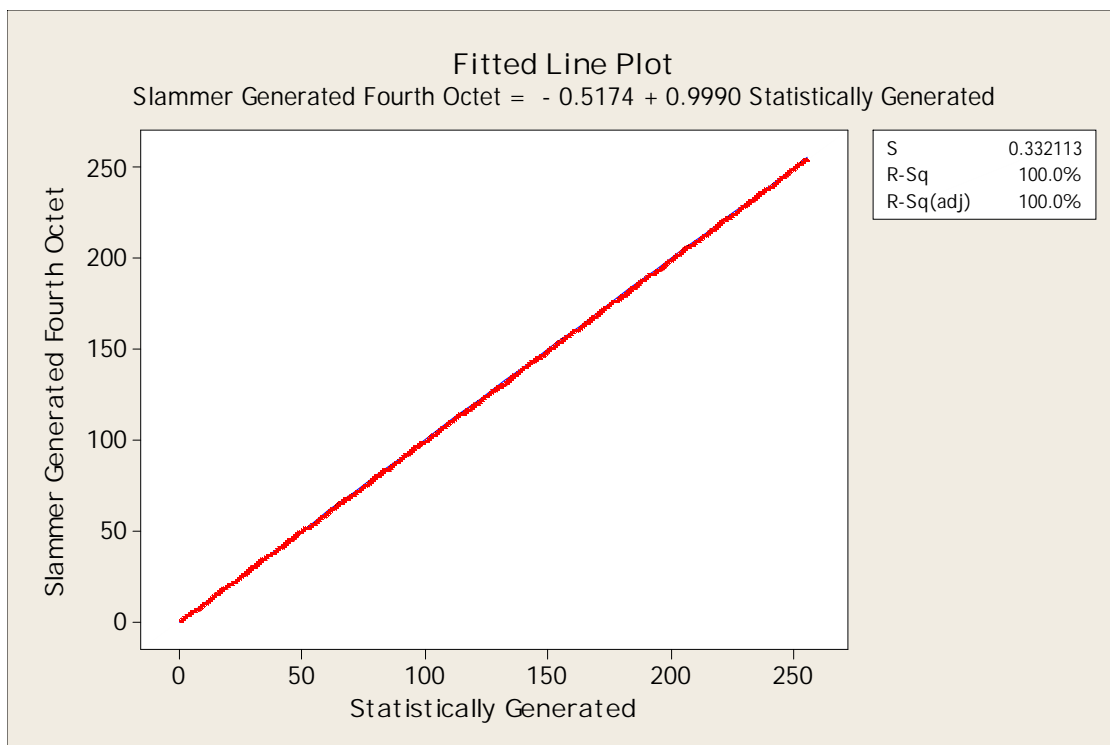


Figure 23. Fitted Line Plot for Slammer-Generated Fourth Octet

From the data presented above, it is shown that the Slammer-generated third and fourth octets are uniformly distributed across the octet range. The statistically generated model and the Slammer-generated packets match to a point of virtual identicalness. This proves that within the total range of IP and octet addresses, Slammer generates a uniformly distributed set of random numbers for packet infection despite the pattern anomaly observed in the lag plots.

### **4.3 Matlab Model Simulation of Slammer Routing Worm Operation**

This first section compares the results of the Matlab model-generated infection rates compared to the results presented by Zou to validate the Matlab model simulation. It also compares the estimated infection doubling rate during the original Slammer worm release versus the Matlab model infection doubling rate. Finally, this section compares the Matlab model infection rate data from this experiment against the infection rate data generated in the Wei research.

#### **4.3.1 Validation of Matlab Model Infection Rate Simulation**

A short visual comparison of the infection rates of the Code Red and routing worms generated by Zou versus the Matlab model infection rates of those worms is provided. A more detailed statistical comparison of the infection rates of original Slammer worm and the “/8” routing worm is compared against the infection rate generated by the Matlab model.

##### **4.3.1.1 Code Red versus Routing Worms**

Figure 24, the Code Red and Zou Code Red routing worm infection rates generated by the Zou research and the infection rates generated by the Matlab model. These infection rate models set the vulnerable systems at 360,000 and scan rate of 358

packets per minute for each worm simulated [ZTG05]. Due to the size of the simulation and the detailed analysis provided of the Slammer infection curves, these graphs are presented as a visual example of the accuracy of the Matlab model of the infection rate. The two charts show a similarity that indicates a strong correlation between the two simulation models. These visual and numerical similarities based on the visual comparisons of the two graphs validate the Matlab model of infection rate. This research was limited to visual comparisons due the unavailability of the raw research data generated by Zou.

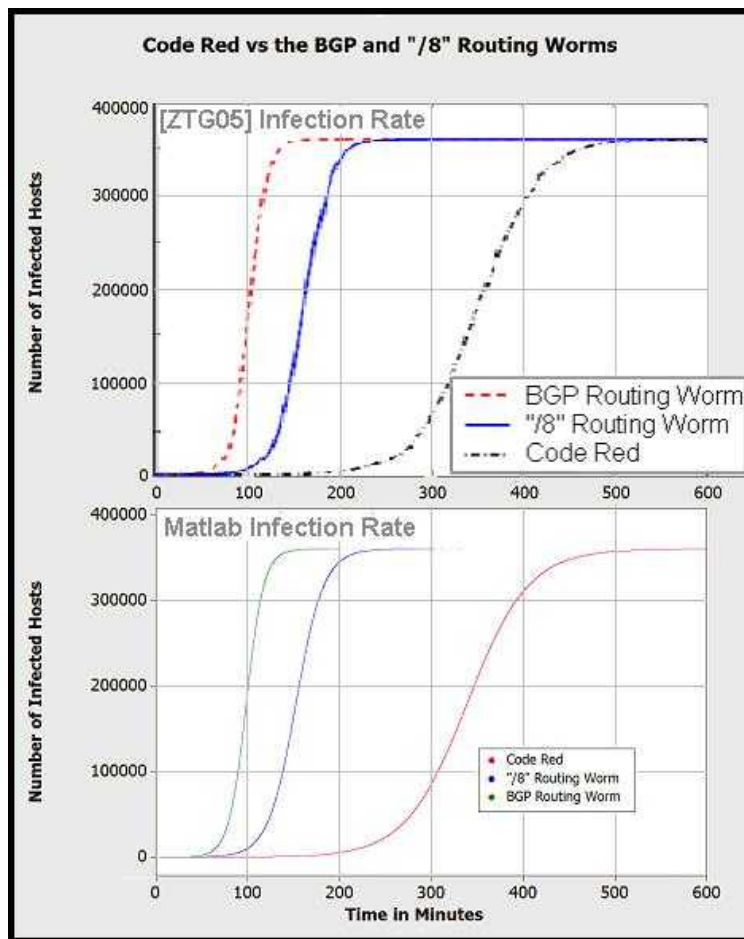


Figure 24. Zou Code Red versus Matlab Model Code Red Infection Rates

#### 4.3.1.2 Original Slammer versus “/8” Routing Worm

The infection rate data for the original Slammer worm is based on an average packet per second rate of 4,000 with 100,000 vulnerable systems [ZTG05]. The number of vulnerable systems, 100,000, appears to be an arbitrary number chosen by Zou for their research as it does not match the observed number of 74,856 systems infected [MPS03]. According to their research, the Slammer routing worm was based on 3,108 pps with the same 100,000 vulnerable systems. The infection rate characteristics found in their research are shown in Figure 25 [ZTG05].

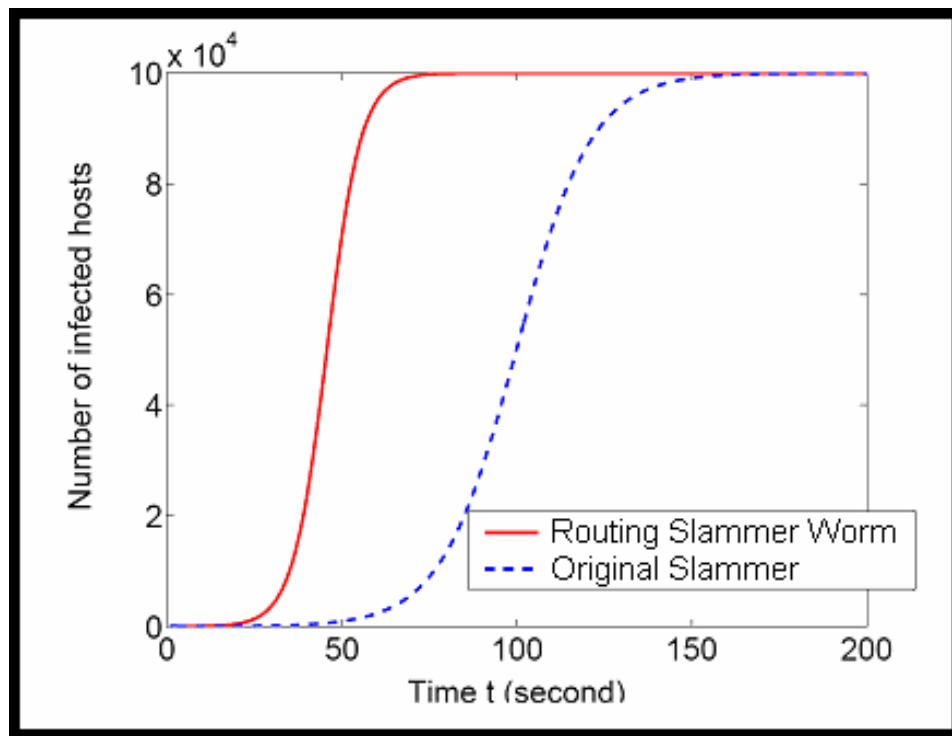


Figure 25. Zou Slammer Worm versus Zou Slammer Routing Worm

The infection rate generated by the Matlab model for comparison to the Zou research is run 20 times and a 95% confidence interval is provided. Each of these infection rate curves uses the same factors of 100,000 vulnerable systems and an initial number of infected systems of 10 hosts used by Zou [ZTG05]. The Matlab model



generation of the Slammer routing worm infection curve is generated using the 3,108 pps to account for the increased size of the infection packet [ZTG05]. For the Matlab model generation of the original Slammer worm infection rate, the average of 4,000 pps observed during its original release is used. The two infection rate curves are presented in Figures 26 and 27. Additionally, a third Matlab model-generated infection rate curve for the Slammer routing worm is presented in Figure 28 with a 95% confidence interval over 20 runs with the packets per second set at 4,000. The average infection rate curves from the data shown in Figures 26, 27 and 28 are combined into one graph presented in Figure 29 to show the speed differences between the Matlab models.

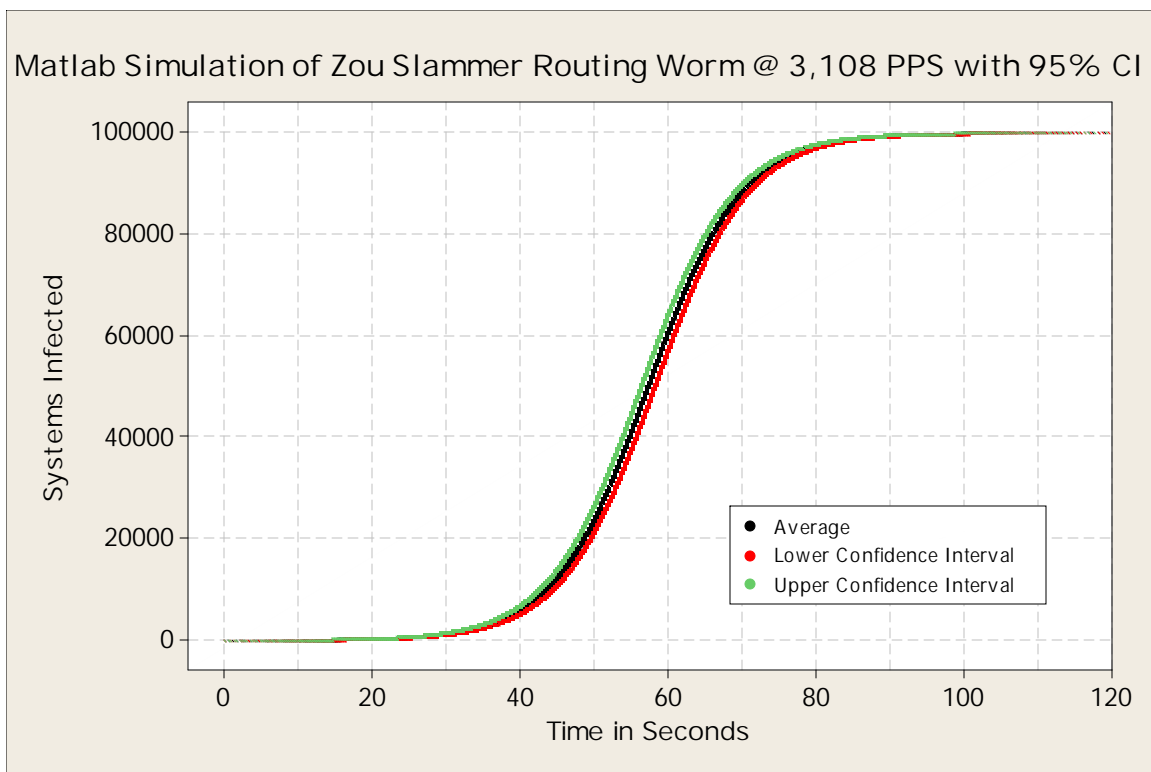


Figure 26. Matlab Model of Zou Slammer Routing Worm @ 3,108 pps

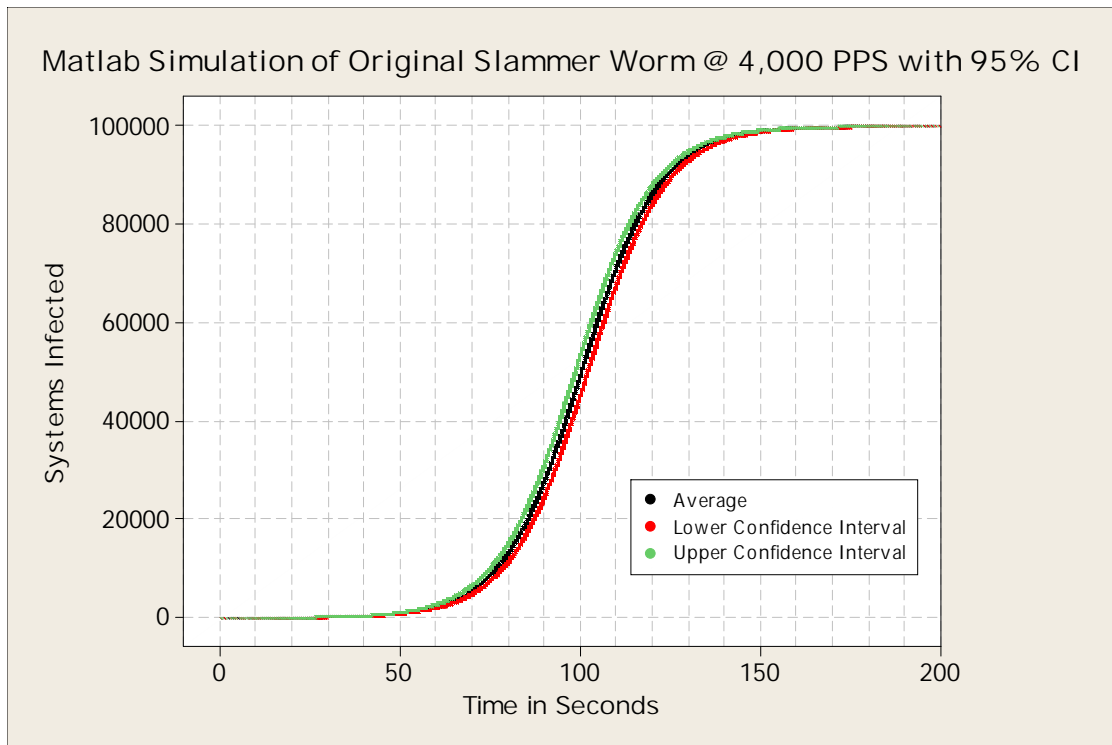


Figure 27. Matlab Model of Original Slammer Worm @ 4,000 pps

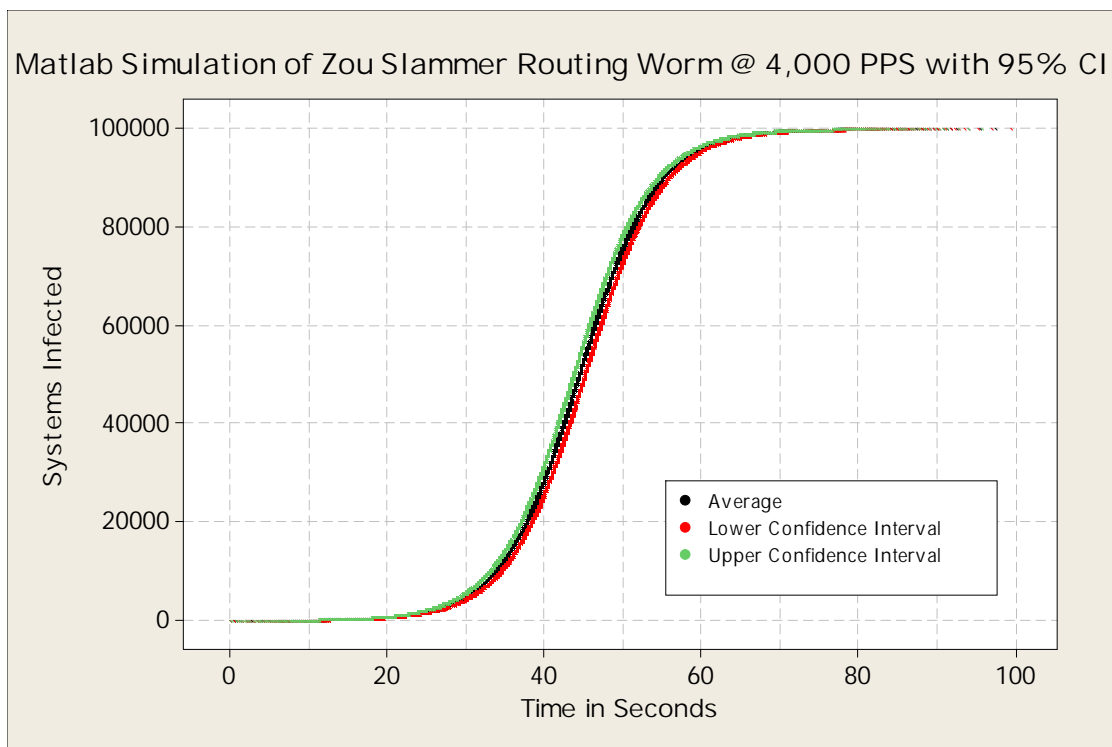


Figure 28. Matlab Model of Zou Slammer Routing Worm @ 4,000 pps

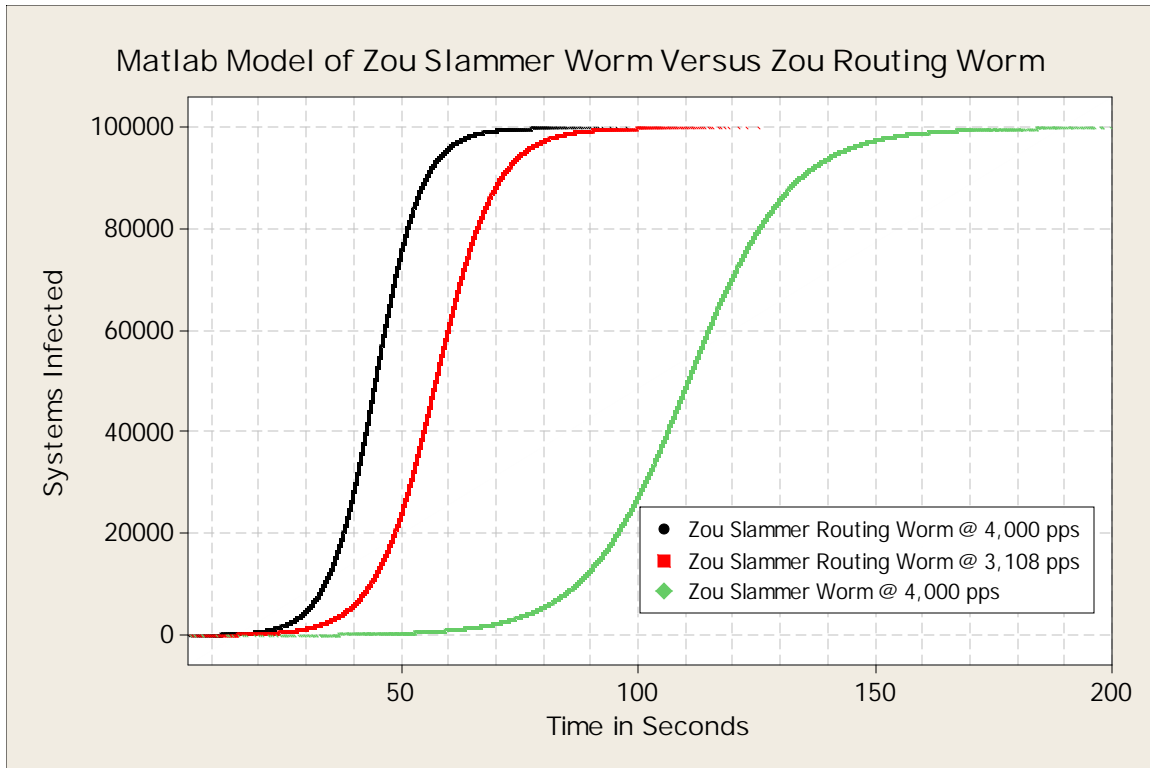


Figure 29. Matlab Model Composite of Slammer Infection Rates

As shown in Figure 30, the infection rate curve of the Slammer routing worm presented by Zou and the Matlab model with the 3,108 pps do not match. However, the Zou Slammer routing worm infection rate curve and the Matlab model with 4,000 pps show a strong similarity. Based on this experiment, the simulation of the 3,108 pps Slammer routing worm has a slower infection rate than the curve generated by the Zou research. Therefore, the simulations shown in the graph by Zou were either completed at the 4,000 pps, the description accompanying the graph is in error, or the infection rate data generated was in error.

Using the validation from the Code Red and routing worm simulation graph comparison and the similarities shown in the original Slammer worm infect rates, the Matlab model-generated infection rate is validated. Furthermore, the observation that the

Slammer routing worm infection rate presented by Zou was performed at the 4,000 pps level provides one additional point of validation of the Matlab model simulations while showing the graph in the Zou paper is in error.

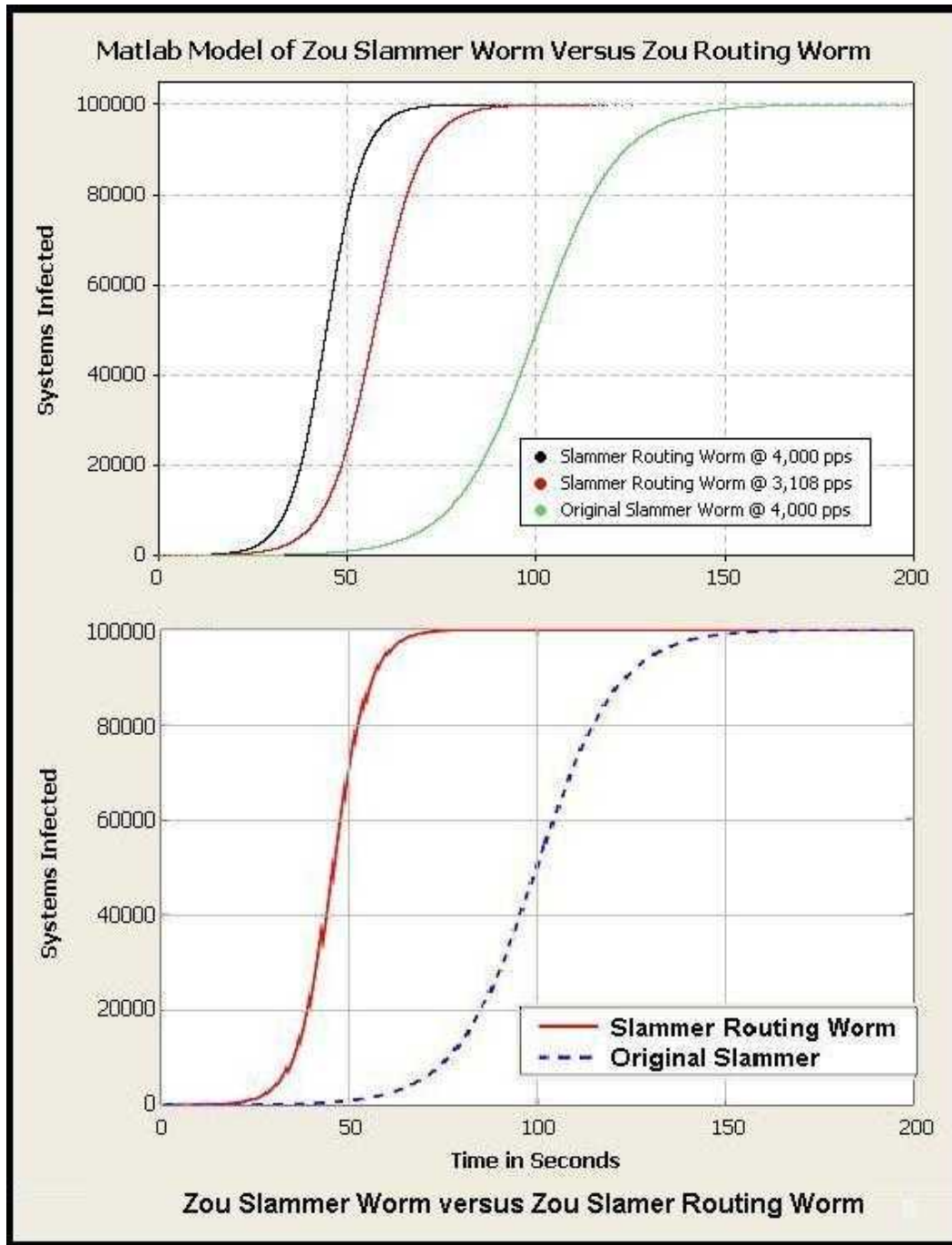


Figure 30. Matlab Model Slammer Worms versus Zou Slammer Worms

#### 4.3.2 Matlab Model of Doubling Rate versus Observed Slammer Rate

The next step in validation of the Matlab model is to compare the Matlab model simulation of the full Internet model with the actual numbers of Slammer vulnerable systems against the observed rate estimated during Slammer's original release. For comparison to the "real world" observations, this experiment uses the standard 74,856 vulnerable systems with an average of 4,000 pps [MPW03]. Moore observed that a single worm had the potential to infect 7 (+/- 1) vulnerable systems per second [MPW03]. This translated to a global doubling rate of 8.5 (+/- 1) seconds which is used to generate a doubling rate curve with upper and lower bounds set at 9.5 and 7.5 seconds respectively [MPW03]. Within the Moore initial report, there is mention that this doubling rate was calculated for the first minute [MPW03]. Each of the doubling rate graphs presented below show the extension of that doubling rate to  $2^{16}$ , or 65,536 systems infected, which is the largest doubling factor prior to exceeding the total number of vulnerable systems. For clarity, a one minute reference line is provided on each of the infection doubling rate graphs to illustrate the cut-off of the one minute estimate by Moore [MPW03].

The Matlab model-generated doubling rate is presented in Figure 31 using, as mentioned above, the original Slammer worm average of 4,000 packets per second as observed in 2003. The confidence intervals are set at 95% for the Matlab model-generated Slammer infection doubling rate. The average doubling rate generated by the Matlab model at a 95% confidence interval is 9.348 (+/-0.763) seconds for the first minute. This is a difference of 0.848 (+/-0.237) seconds from the estimate of the original Slammer worm doubling rate. The estimate of the original Slammer worm infection

doubling rate with the upper and lower bounds is presented in Figure 32 providing a curve representing the estimated infection rate. Then finally for ease of comparison, the estimated doubling rate of the original Slammer worm and the Matlab model-generated doubling rate of the Slammer worm are plotted onto the same graph in Figure 33.

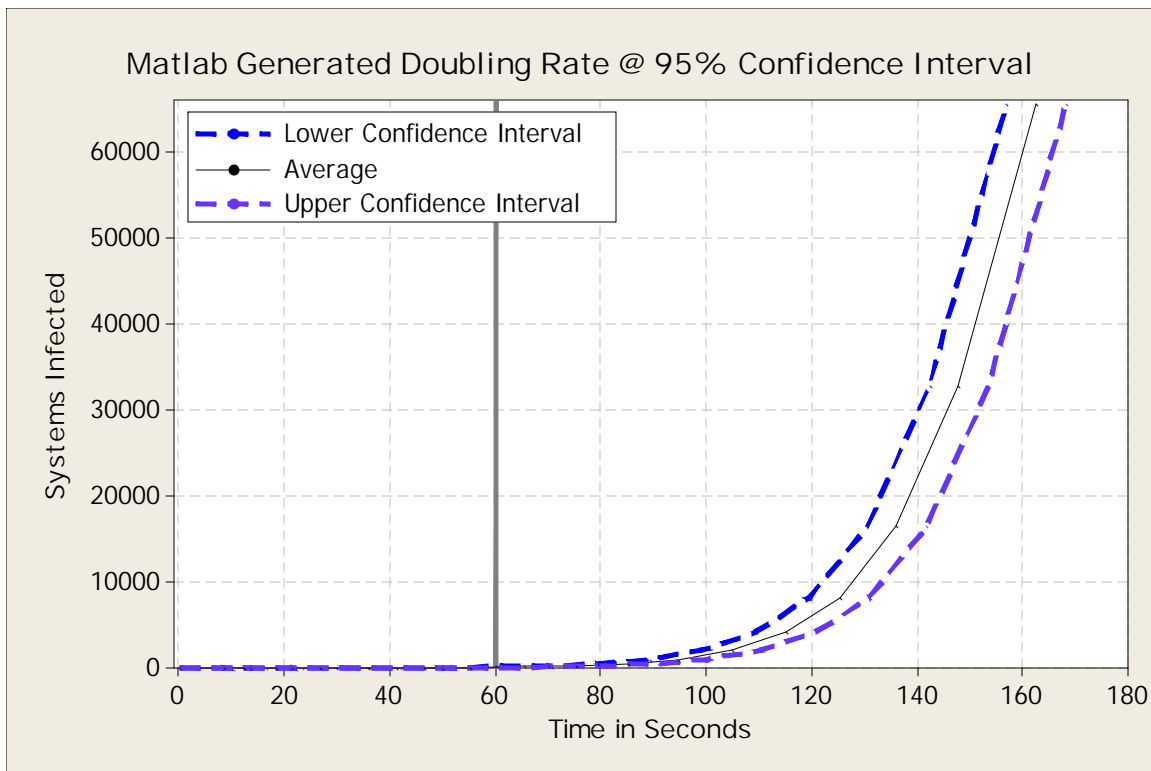


Figure 31. Matlab Model-Generated Doubling Rate

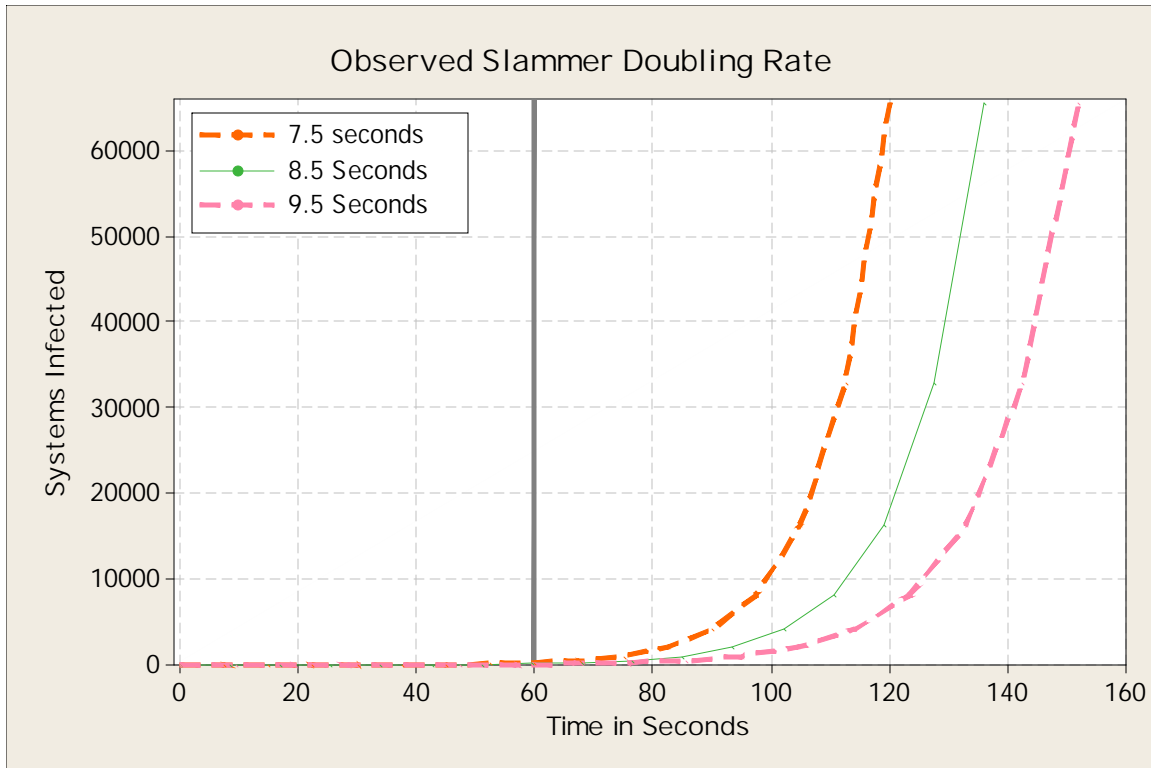


Figure 32. Slammer Doubling Rate

As the combined data shows in Figure 33, the estimated doubling rate of the original Slammer and the Matlab model-generated doubling rate overlap for just over 140 seconds. This extension of the doubling rates beyond the one minute limitation shows that when continued to their infection limits, the two rates eventually separate and shows the original Slammer doubling rate completes the infection of the remaining systems faster than the Matlab model-generated doubling rate. This indicates that if the original Slammer doubling rate remained within the bounds set by the original estimate through the infection of the 65,536<sup>th</sup> system, the Slammer worm was faster than .

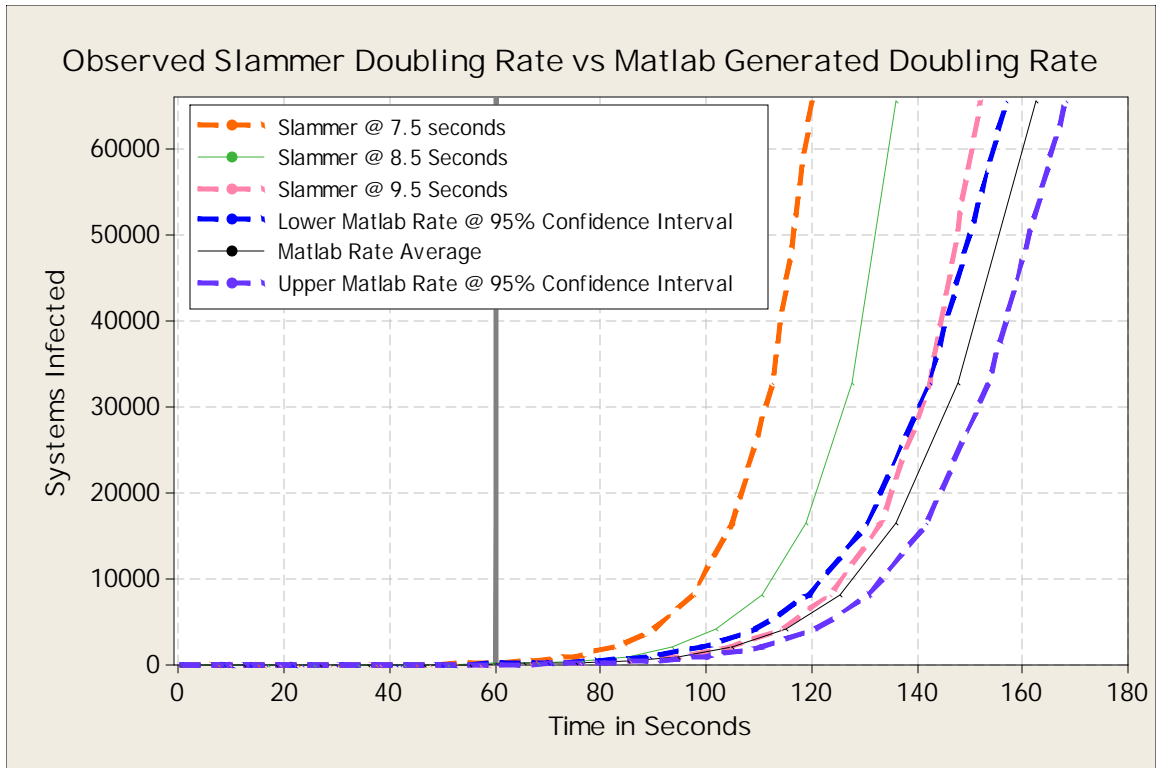


Figure 33. Matlab Model versus Observed Slammer Doubling Rate

The closer view isolating the one minute limitation, shown in Figure 34, shows that there is some significant overlap of the two doubling rates. The upper limit of the original Slammer doubling rate average is contained within the lower confidence interval of the Matlab model-generated doubling rate. The containment of the observed Slammer doubling rate within the Matlab model until the 60-second point provides a further indication that the Matlab model is valid. Further, the average of the original Slammer worm doubling rate was faster than the average that could be expected if the original Slammer worm was released more than once.



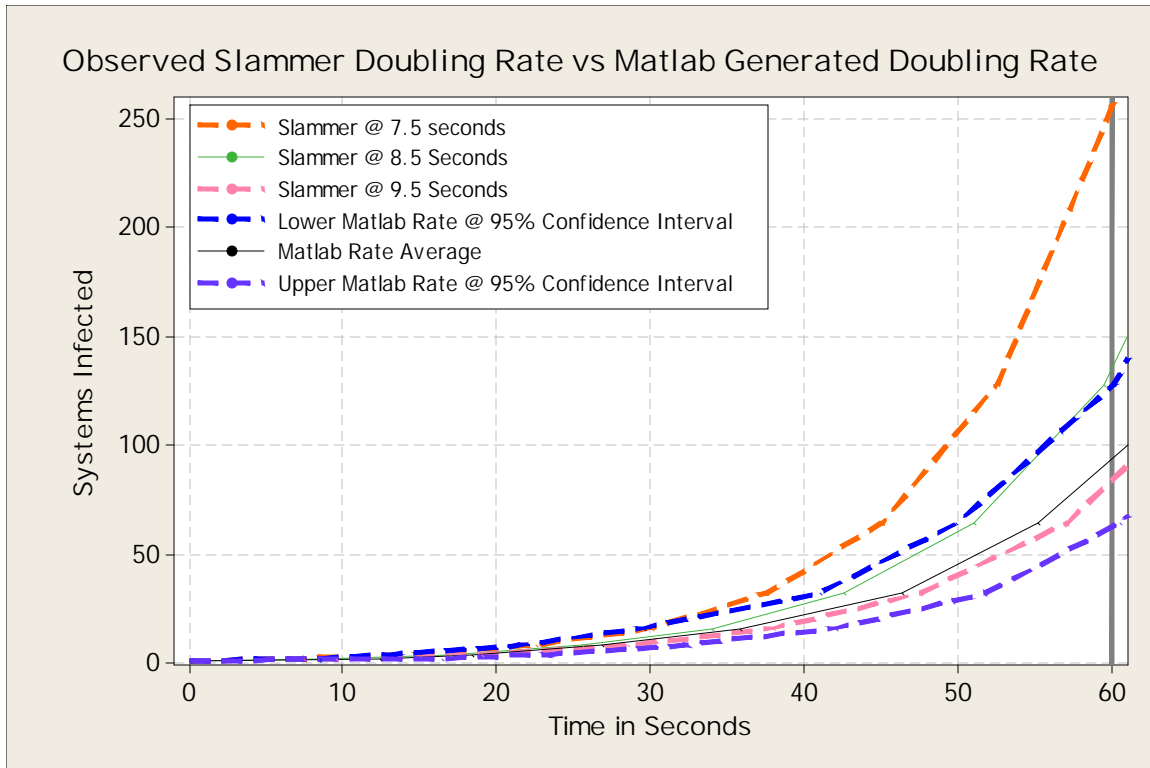


Figure 34. Matlab Model versus Slammer Doubling Rate Detailed

Based on this experimental data, it is apparent that original Slammer worm operated faster than the expected average case determined by the validated Matlab model, to a point that it was always faster than the Matlab average and only contained inside the upper (i.e., fastest) confidence interval at 95%. Thus, the research shows that the original Slammer worm doubling rate, when originally released, operated faster than could be expected with multiple instances of its release.

From the validation of this Matlab model by the estimated infection doubling rate and based on the previous validation of the Matlab model by the comparisons to the Zou research this model is validated using two methods. Therefore, the full Internet infection rate of the Slammer Worm as calculated by the Matlab model simulations, and shown in

Figure 35, can be considered a valid model of how the Slammer worm would operate on a computing system in 2003.

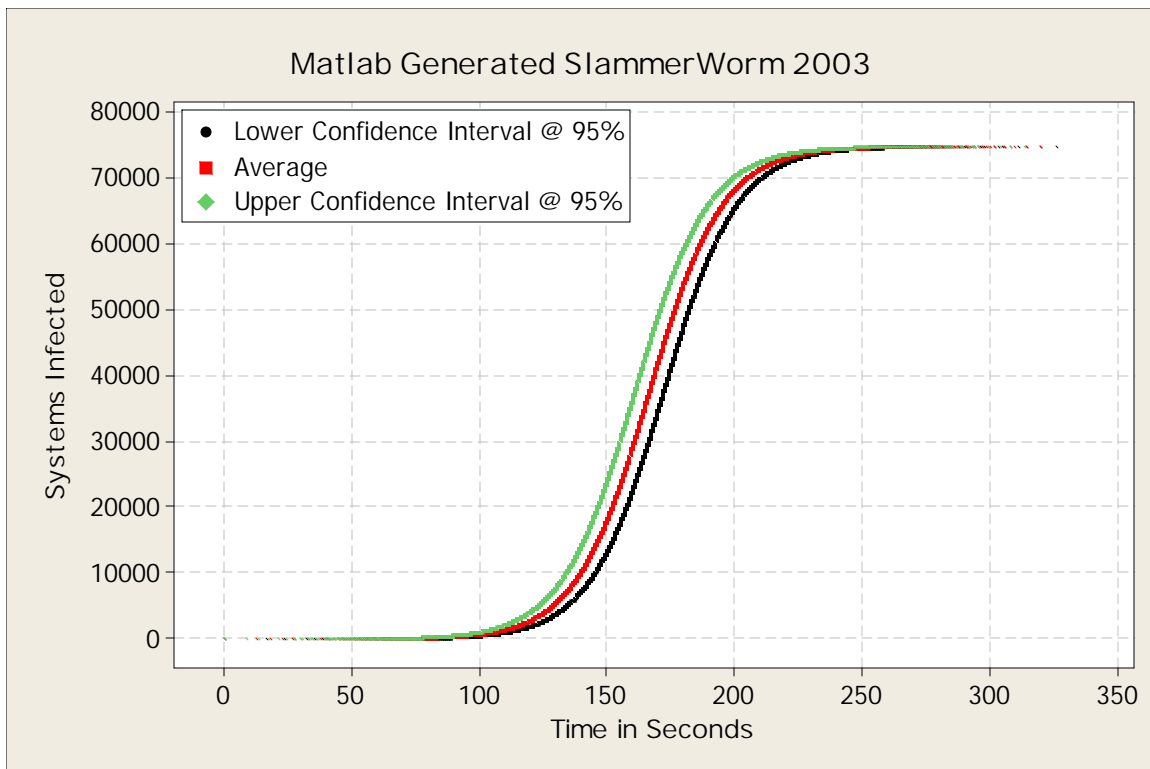


Figure 35. Matlab Model-Generated Slammer Worm 2003

This infection rate for Figure 35 is calculated using 74,856 vulnerable systems with the original Slammer worm average of 4,000 pps at a 95% confidence interval over 20 runs. The data shows that over 70,000 vulnerable systems would be infected between 198.825 and 210.078 seconds with a 95% probability. This is over 93% of the potential victims and it is well under the ten minute estimated during the original observation of the Slammer worm infection. This estimate was calculated by using the number of scans observed at the three minute point of Slammer's original release and extrapolating the expected time to scan 90% of the address space [MPW03]. Thus, this ten minute estimate was not an actual measurement of systems infected at the ten-minute point.

The conclusion that can be reached from this data is that while the Slammer infection doubling rates are not exactly the same, the first minute doubling rates do show significant overlap. The doubling rates are both exponential and only differ in their estimation of the rate of infection.

The differences between the estimated time to infect 90% of the systems and the Matlab model simulation in expected time for reaching the 90% level of infected systems is larger than expected. However, there are several reasons that can be given as to why this variation in the two data sets occurred. These include the admission by Moore in their research that not all of the data sets they analyzed were sufficiently precise over that initial short collection duration, which may have affected their analysis of how fast the doubling rate occurred [MPW03]. The difficulties of collecting accurate data during the original Slammer worm release were further exacerbated by an unexplained transient failure at the 2 minute and 40 second point after Slammer's release [MPW03]. The ability to repeat the Matlab model simulation of the Slammer worm infection rate provides a database from which to draw a more comprehensive statistical model than does a single observation of the Slammer worm in the wild. This means that the data estimates used by Moore to generate their results are but a single instance of how Slammer acted. Slammer's behavior would almost certainly have been different given other releases.

#### **4.3.3 Matlab Model versus Wei Slammer Infection Rate**

The Slammer experiments run by Wei, covered how the Slammer worm would react with differing traffic loads and network failures. Their research included a simple baseline test with the 75,000 vulnerable hosts and 4,000 scans per second observed in

2003. The baseline curve was presented with several other curves that showed the Slammer worm operation with varying traffic loads and network failures as shown in Figure 36 [WMS05].

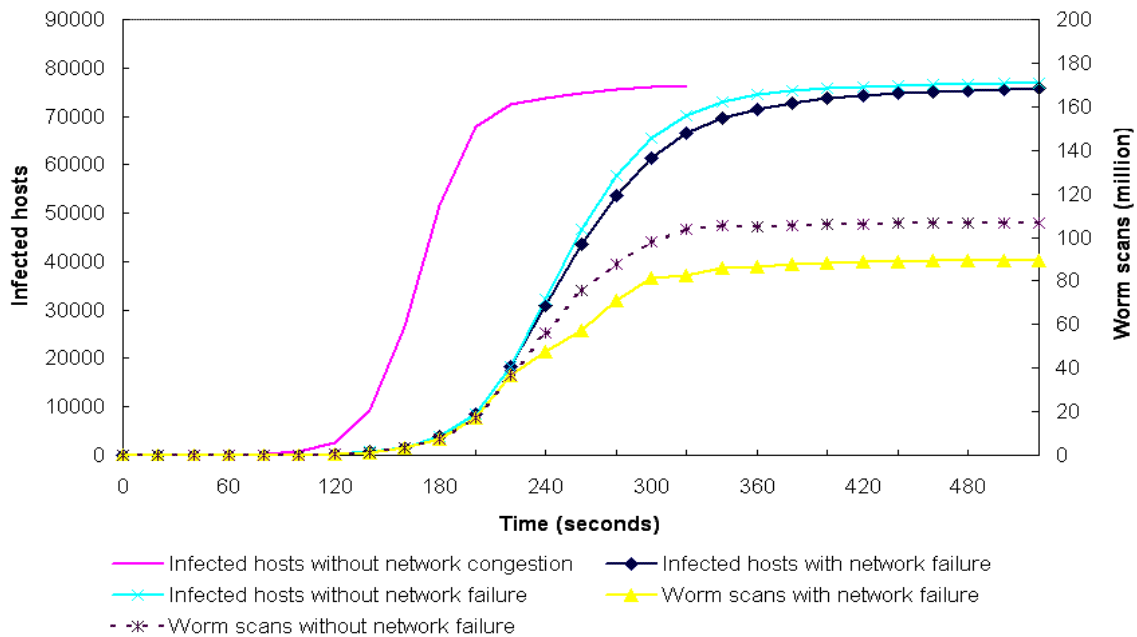


Figure 36. Wei Slammer Worm Simulation

The Slammer infection rate generated by the Wei research closely matches the curve generated of the Slammer worm infection rate by the Matlab model in this research as shown in Figure 37. The Wei experiment follows the Matlab model's lower confidence interval through 50,000 systems infected. The lower confidence interval of the Matlab model continues to closely match the Wei data until just past 65,000 systems infected where Wei's data makes an uncharacteristic deviation from a smooth curve. Despite this top end deviation, the Wei data further supports the Matlab model of the Slammer worm infection rate as an accurate representation of how the Slammer worm behaves in the wild.

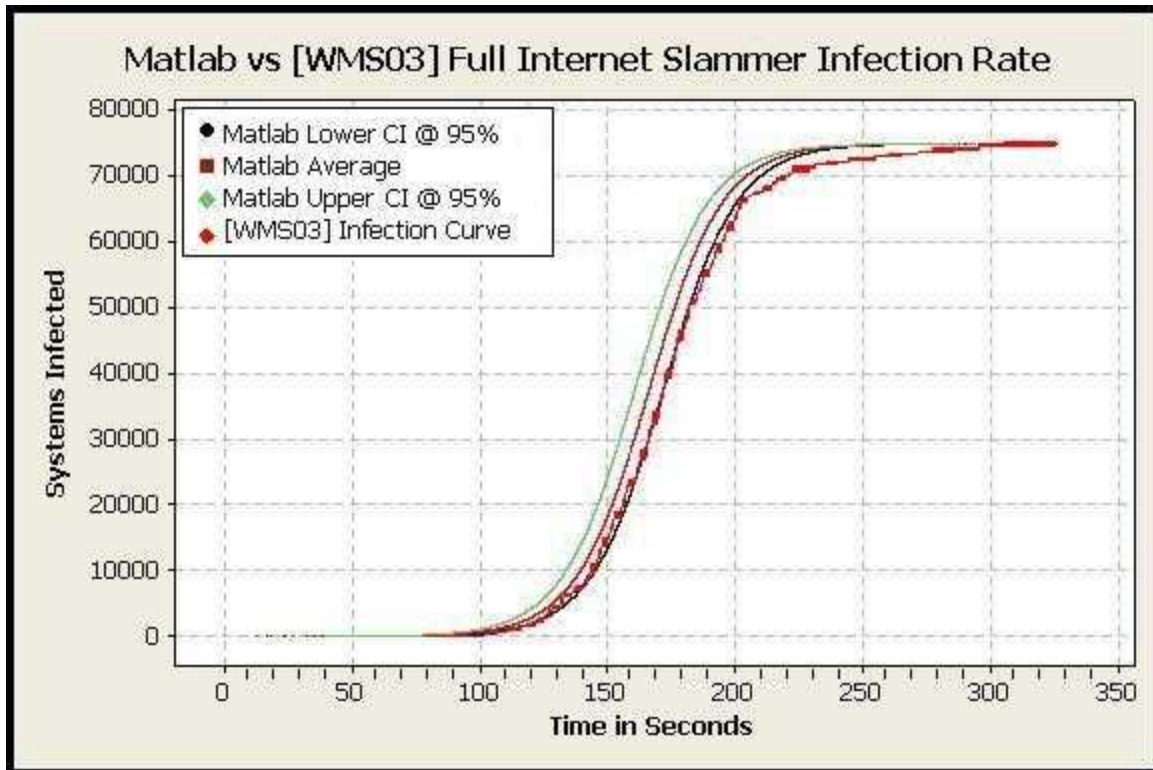


Figure 37. Matlab Model Slammer Worm versus Wei Slammer Worm

#### 4.4 Scanning Worms in a Computing Architecture of Today

Zou's research used an arbitrary number of Slammer vulnerable systems set at 100,000 [ZTG05]. The actual number of potential victims is 74,856 [MPS03]. This section takes the Zou experiment a few steps further by using the correct number of vulnerable systems and analyzes the operation of the scanning worms in a 2003 and 2007 computing environment. Thus the number of vulnerable systems is set at 74,856 and each of them are run with a 95% confidence interval and displayed separately.

Note however, that although the speed increase of the network alone is considered here in this experiment, the increase in the number of vulnerable systems is not. For ease of comparison, the vulnerable systems were left the same as any increase in the number of potential victim systems reduces the non-vulnerable systems by an equal number.

Thus, for every vulnerable system added above the 74,856 threshold the worm will spread faster. For example, if the original Slammer worm using the 100,000 vulnerable systems as used by Zou is compared to the infection rate using 74,856 vulnerable systems where both are set at the average 4,000 pps as observed during the original Slammer worm release [MPW03], the Zou infection rate is faster as shown in Figure 38. Finally, because there is no basis of comparison in this research for the possible number of vulnerable systems as there is for the speed of the scanning packets generated by the Slammer code, the increase in vulnerable systems today is not be considered.

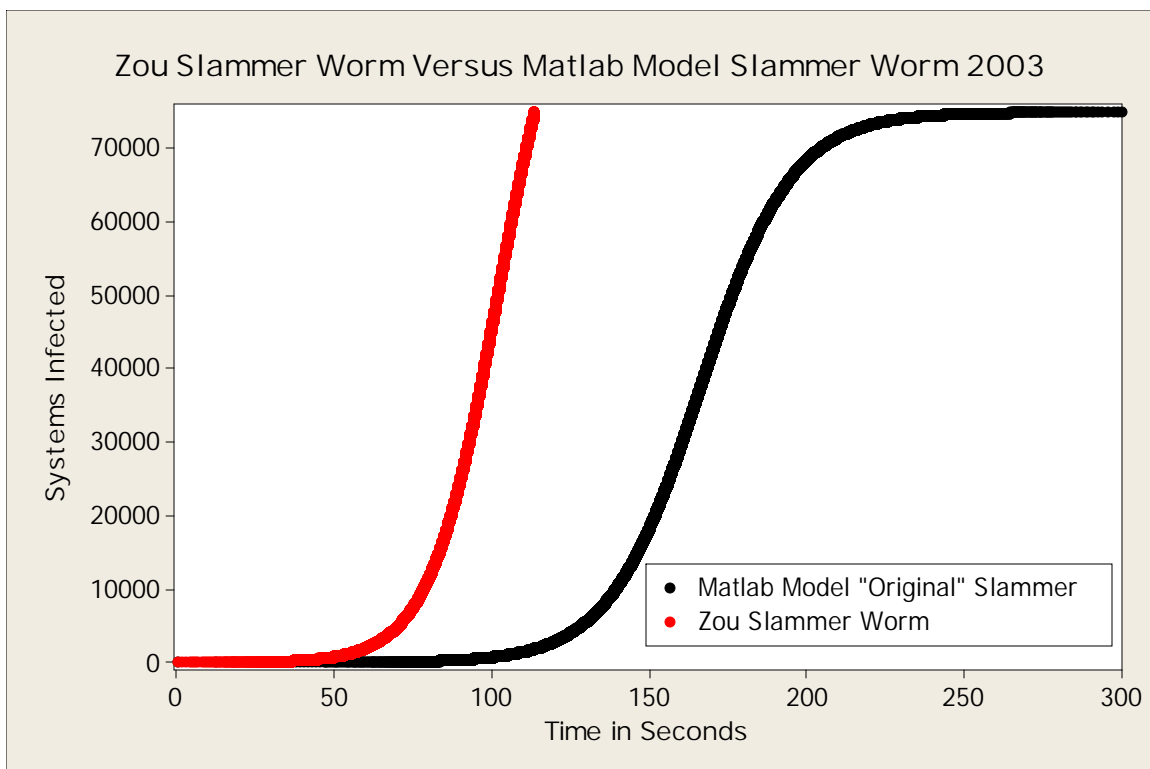


Figure 38. Zou Slammer Worm versus Matlab Model Slammer Worm 2003

To provide a baseline for comparison, a Matlab model of the Slammer worm infection rate average with an upper and lower confidence interval at 95% as it would act in 2003 is provided in Figure 39. This Matlab model uses 4.3 billion available addresses

in the scanning space, 74,856 vulnerable systems, and the average 4,000 pps for infection rate curve generation. Hereafter, this worm is called the Slammer worm 2003.

The Matlab model of the average infection rate curve with an upper and lower confidence interval at 95% of the Slammer worm as though the worm were released on a system today is provided in Figure 40. Using the 4.3 billion available addresses, the 74,856 vulnerable systems, and an average of 14,398 pps found in this research the Matlab model of the infection rate curve is generated. Hereafter, this worm is referred to as the Slammer worm 2007.

The Matlab model average of the Slammer routing worm as it would have acted in 2003 (Hereafter, the Slammer routing worm 2003) is presented in Figure 41 with an upper and lower confidence interval at 95%. The Matlab model that generates this infection rate curve uses 1.95 billion available addresses, the number of vulnerable systems set at 74,856, and 3,108 pps as noted in the Zou research for the larger infection packet size [ZTG05].

In Figure 42, the Matlab model used to generate the average with an upper and lower confidence interval at 95% of the 2007 version of the Slammer routing worm uses a rate of 11,187 pps to reflect the increase for operation of the worm on a computing system of today. The number of vulnerable systems is set at the same 74,856 and 1.95 billion addresses available for scanning as the Slammer routing worm 2003. Hereafter, this worm is referred to as the Slammer routing worm 2007.

Finally, the average infection rate of the Slammer worm 2003, Slammer worm 2007, Slammer routing worm 2003, and Slammer routing worm 2007 are plotted onto one graph for ease of speed comparison in Figure 43. This graph shows that the infection

rates of worms on a system of today are faster than their 2003 counterparts. Table 18 shows the variable settings for all of the 2003 and 2007 worms.

Table 18. Matlab Model Variables for 2003 versus 2007 Worm Comparison

	Number of IP addresses	Number of Vulnerable Systems	Number of Iterations	Number of Trials	Initial Number of Infected Systems
Slammer Worm 2003	4,294,067,296	74,856	10,000,000	20	One
Slammer Worm 2007	4,294,067,296	74,856	10,000,000	20	One
Slammer Routing Worm 2003	1,946,156,941	74,856	500,000	20	One
Slammer Routing Worm 2007	1,946,156,941	74,856	500,000	20	One

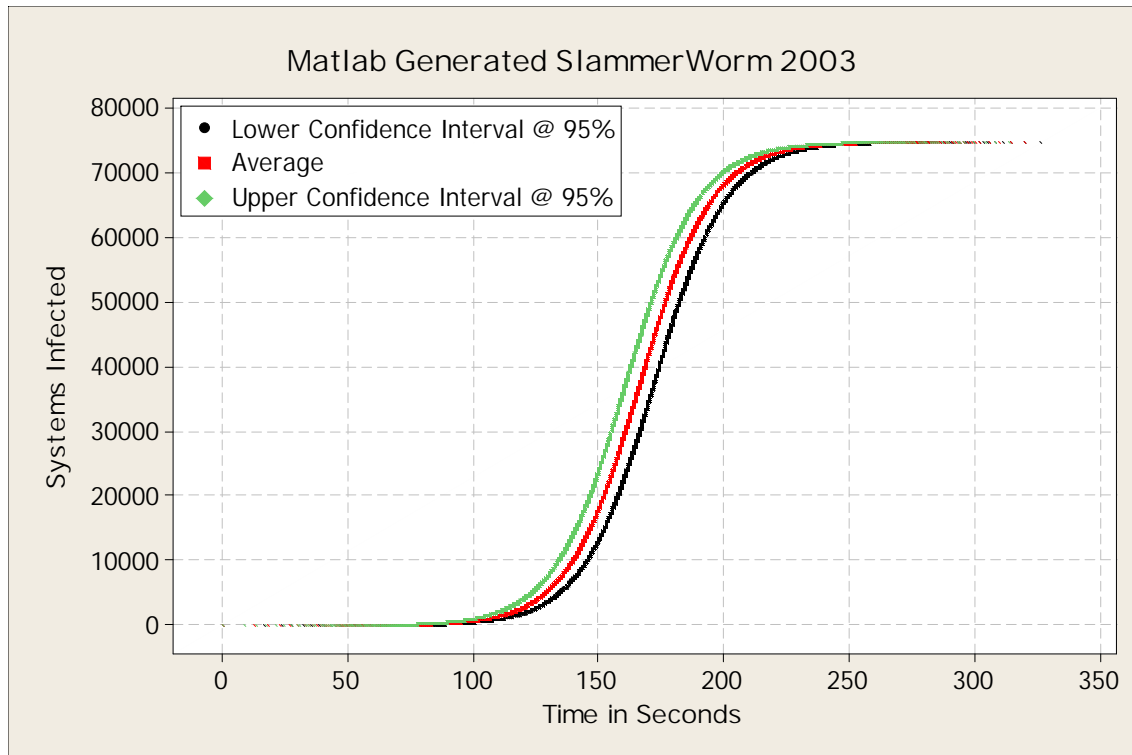


Figure 39. Matlab Model-Generated Slammer Worm 2003



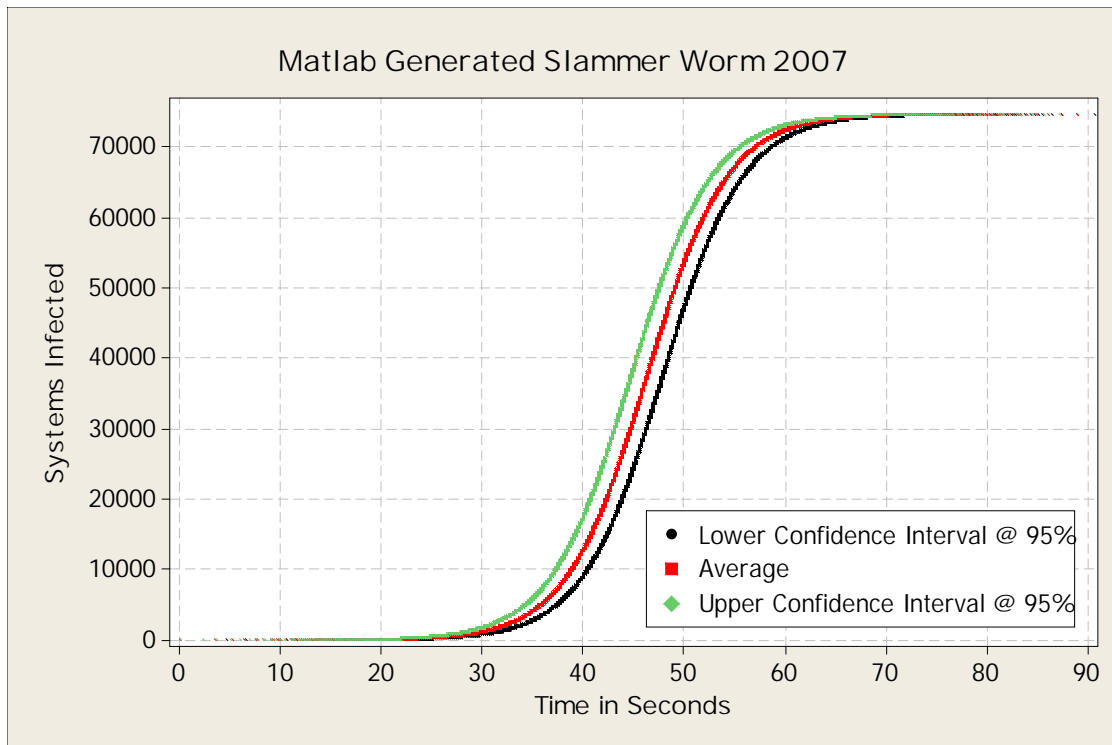


Figure 40. Matlab Model-Generated Slammer Worm 2007

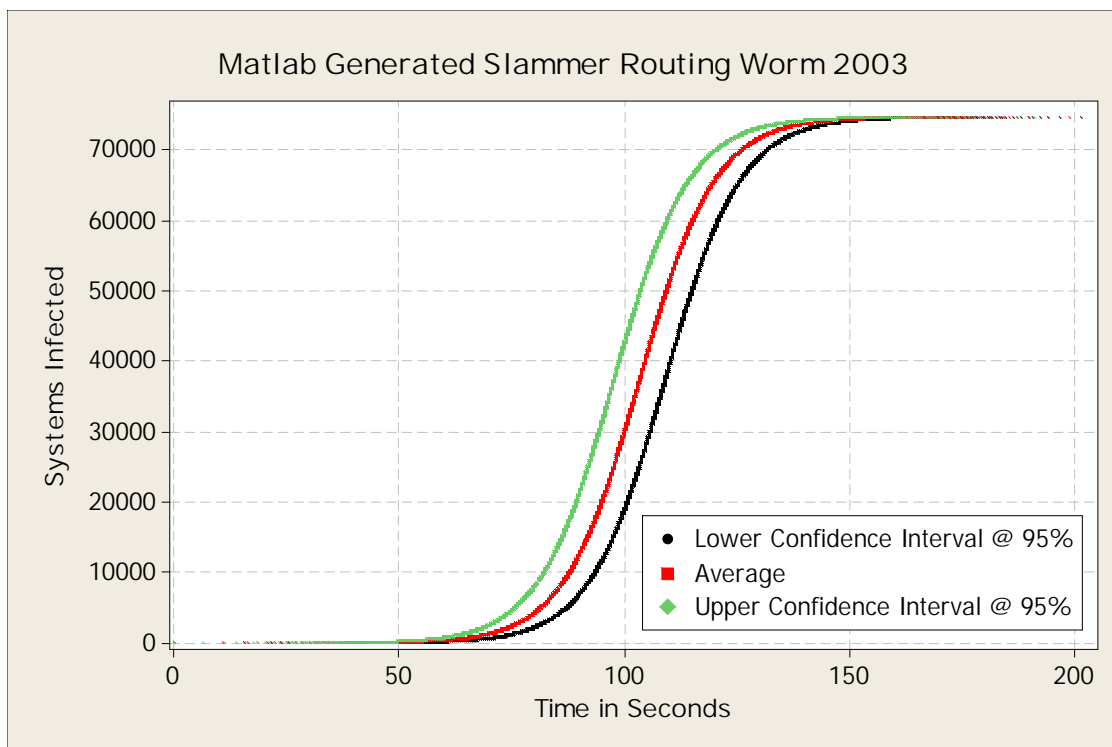


Figure 41. Matlab Model-Generated Slammer Routing Worm 2003

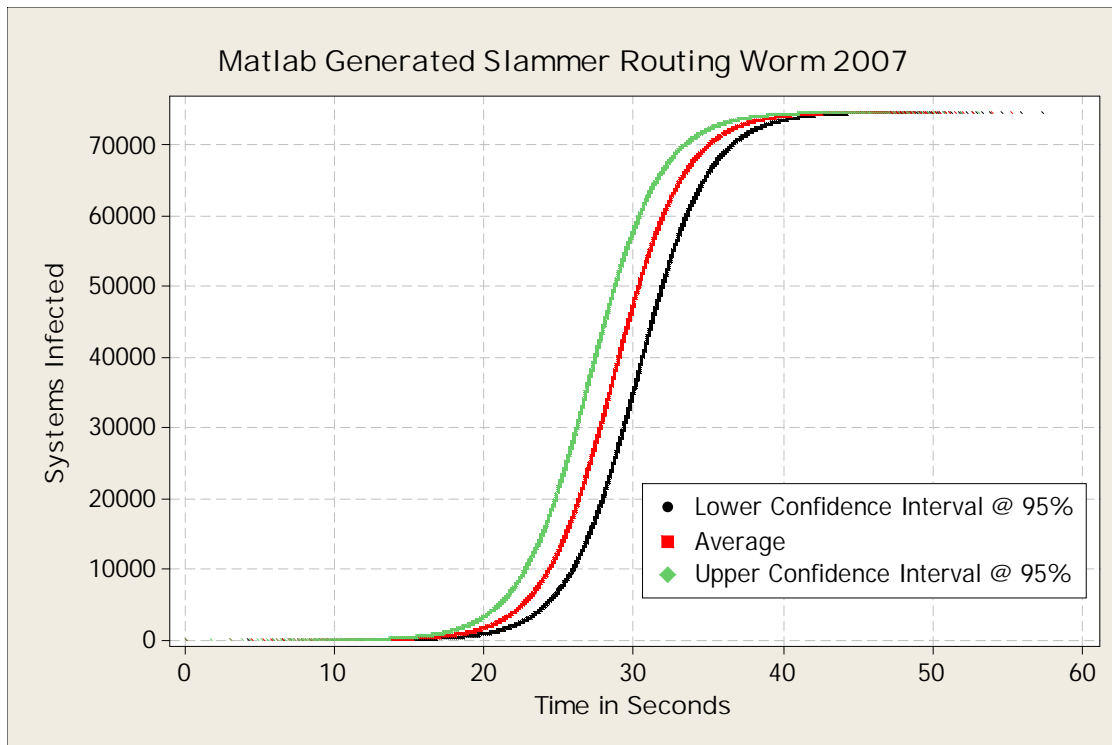


Figure 42. Matlab Model-Generated Slammer Routing Worm 2007

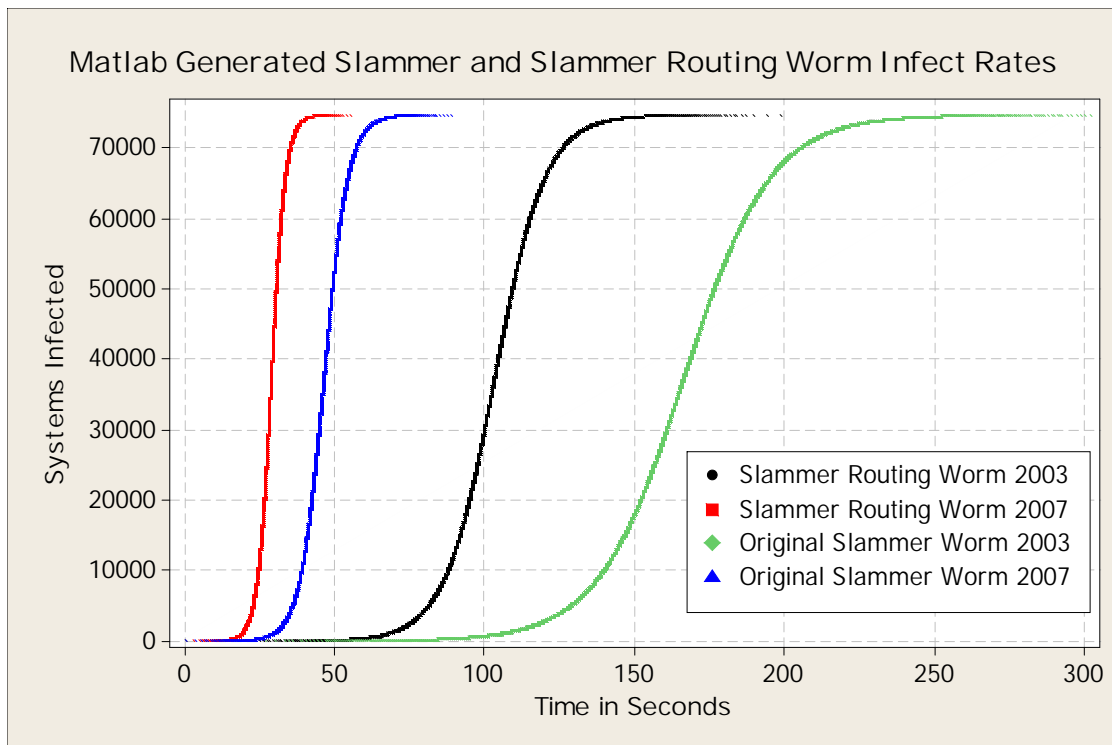


Figure 43. Matlab Model-Generated Slammer Worms and Slammer Routing Worms

This experimental data shows that a Slammer-based worm released onto the Internet of today is much faster than its 2003 counterparts. As shown in Table 19, the increase in the infection of 90% of the potential victims for the two chronologically separated worms are almost identical at 3.599 fold increase. This increase demonstrates the quicker infection rate due to the faster infection packet generation rate.

Table 19. Matlab Model of 2003 versus 2007 Infection Rates

Worm Name	Infection Rate in Seconds			Infection Rate Increase
	Upper CI	Average	Lower CI	
Slammer Worm 2003	203.34	197.69	192.04	3.5995
Slammer Worm 2007	56.49	54.92	53.35	3.5994
Slammer Routing Worm 2003	127.15	121.45	115.76	3.5994
Slammer Routing Worm 2007	35.32	33.74	32.16	3.5994

#### 4.5 Single Slash Eight (SSE) Routing Worm

This section covers the Single Slash Eight (SSE) routing worm and the modeling of its infection rate curve. The final portion of this section provides a comparison of the SSE routing worm against the Slammer worm 2003, Slammer worm 2007, Slammer routing worm 2003, and Slammer routing worm 2007.

##### 4.5.1 SSE Routing Worm Creation

Taking the division of the Slammer routing worm one step further, the original Slammer worm code is modified to become an SSE routing worm which scans only one of the 116 “/8” IANA address spaces. The SSE routing worm is creation details are available upon request

#### 4.5.2 Matlab Model of the SSE Routing Worm

The small changes to the original Slammer worm code discussed in Section 4.5.1 increase the size of the SSE routing worm from 404 bytes to only 412 bytes. Thus, a 2003 version of the Slammer worm with an average of 4,000 pps is changed to a rate of 3,922 pps for the 2003 version of the SSE routing worm. The 2007 SSE routing worm packets per second is barely affected by the addition of only eight bytes and would slow the generation of infection packet on an infected system from 14,398 pps to 14,118 pps.

An IP address space of 16,777,216 possible addresses is used to simulate the address space an SSE routing worm is required to scan with the single CIDR “/8.” Because the total address space is reduced, the total number of vulnerable systems is reduced by equally dividing the 74,856 by the 116 available address spaces. This equates to 645 vulnerable systems in each SSE routing worm range. Table 20 shows the variables used in the generation of the infection rate curves by the Matlab model for the 2003 and 2007 SSE routing worms.

Table 20. Matlab Model Variables for SSE Routing Worm

	Number of IP addresses	Number of Vulnerable Systems	Number of Iterations	Number of Trials	Initial Number of Infected Systems
SSE Routing Worm	16,777,216	645	500,000	50	One

#### 4.5.3 SSE Routing Worm Infection Rate Comparison

Figures 45 and 46 show the 2003 and 2007 SSE routing worm infection rate curves with the accompanying 95% confidence intervals. The 2003 SSE routing worm

infects over 90% of the vulnerable systems in under one minute. However, the 2007 SSE routing worm infects over 90% of the vulnerable systems in under 17 seconds.

The speed increase of these two worms over their 2003/2007 Slammer worm and 2003/2007 Slammer routing worm counterparts is substantial as shown in Figure 47. The SSE routing worm infect rates are aggregated across the entire population to provide a proportional comparison in Figure 47.

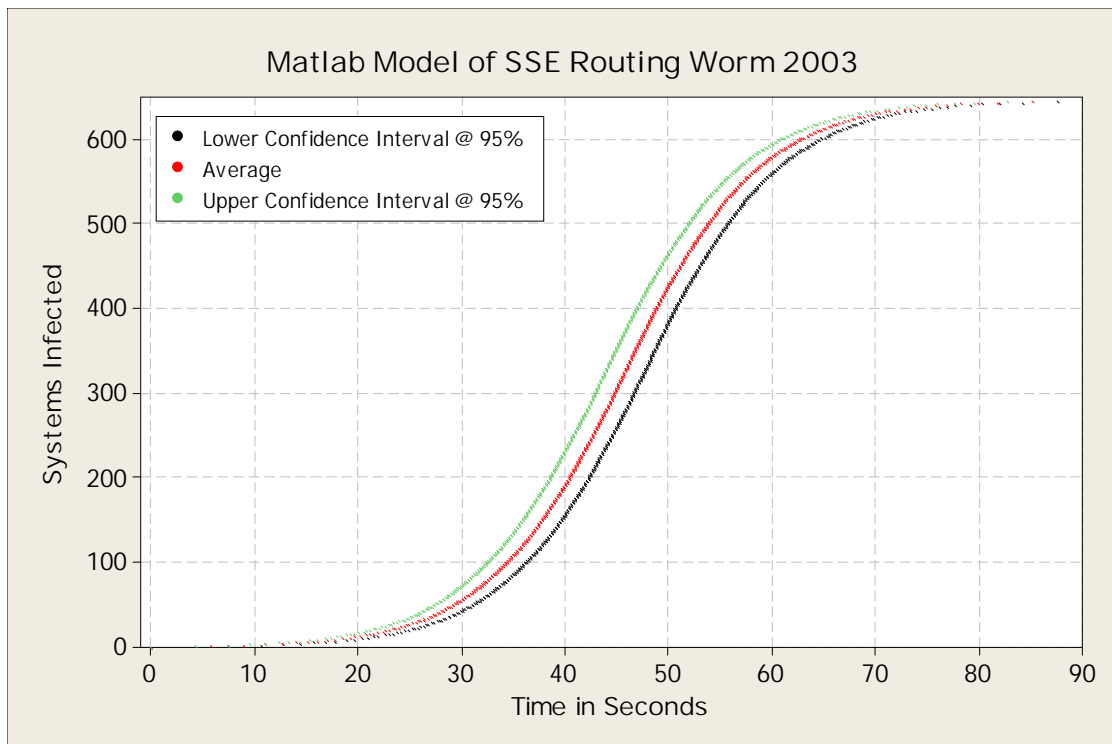


Figure 44. Matlab Model-Generated SSE Routing Worm 2003

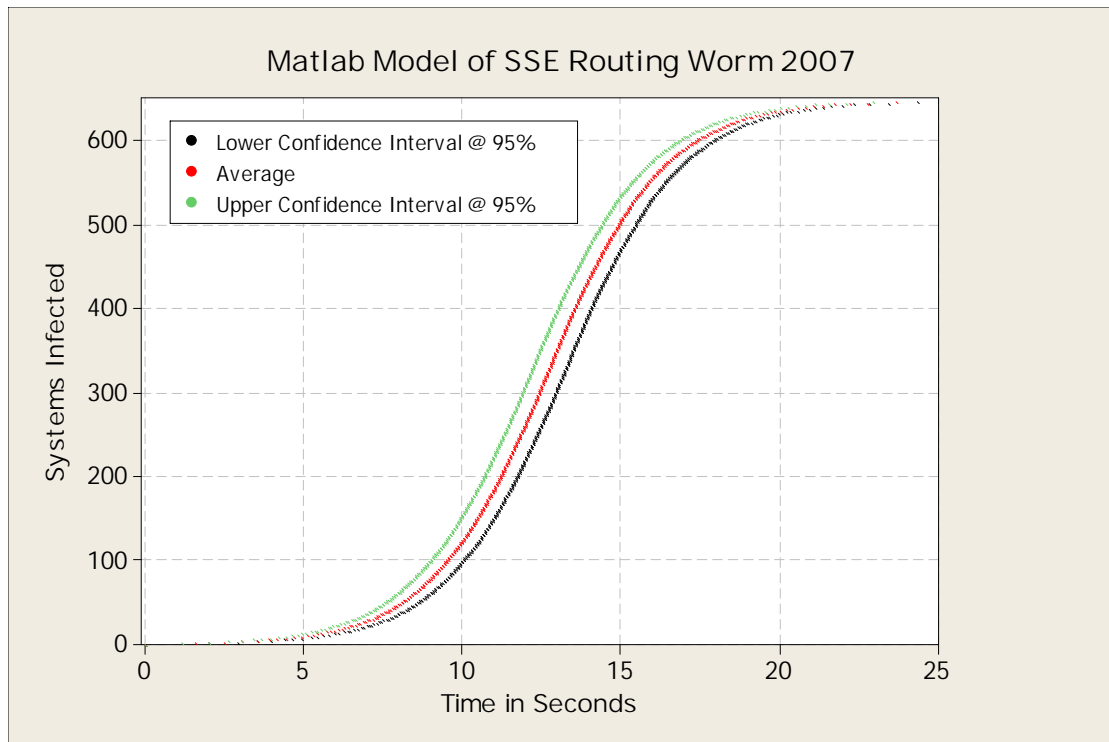


Figure 45. Matlab Model-Generated SSE Routing Worm 2007

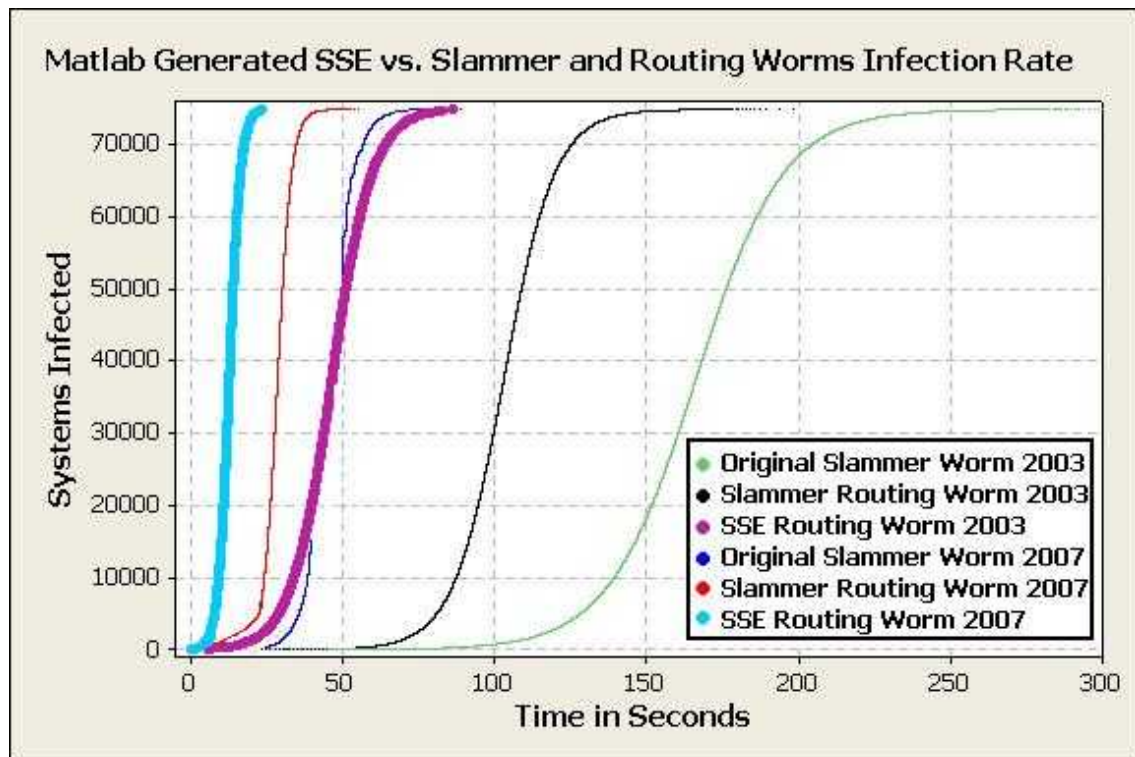


Figure 46. Matlab Model SSE Routing Worms versus Slammer Worms

Table 21 contains the speeds at which these six worms infect over 90% of their vulnerable systems.

Table 21. SSE Routing Worm Speed Comparison

Worm Name	Infection Rate in Seconds		
Slammer worm 2003	Upper CI	Average	Lower CI
	203.34	197.69	192.04
Slammer Routing Worm 2003	Upper CI	Average	Lower CI
	127.15	121.45	115.76
SSE Routing Worm 2003	Upper CI	Average	Lower CI
	61.83	59.92	58.01
Slammer Worm 2007	Upper CI	Average	Lower CI
	56.49	54.92	53.35
Slammer Routing Worm 2007	Upper CI	Average	Lower CI
	35.32	33.74	32.16
SSE Routing Worm 2007	Upper CI	Average	Lower CI
	17.18	16.65	16.11

Overall, the data collected shows that the SSE routing worm was 3.299 times faster than both the 2003 and 2007 Slammer worms. Additionally, the SSE routing worm is also 2.027 faster than the 2003 and 2007 Slammer routing worms. Finally, the increase in infection rate for the SSE routing worm from 2003 to 2007 is 3.6 times due to the increase in infection packet generation rate.

#### 4.6 Summary

The first section of this chapter covered the network configuration for all of the experiments performed in this research and briefly covered the Slammer worm code used as a basis for this experiment. The second section then delved into the results and analysis discovered through this research.

This chapter presents the results and analysis of the data collected from the experiment simulations of the worm infections. Section 4.1 covers the collection and analysis of Slammer's packet per second generation. Section 4.2 examines the randomness of Slammer's IP address and octet generation. In Section 4.3, the Matlab worm models are presented for comparison to the original Slammer worm and the routing worm models proposed by Zou. The comparison of worm speeds possible on computing systems of today versus those available in 2003 is provided in Section 4.4. Section 4.5 examines the infection rate of the original Slammer worm versus the SSE routing worm.



## V. Conclusions and Recommendations

### 5.1 Restatement of the Problem and Conclusions

The primary focus of this experiment was to show that the variety of scanning worms tested were faster than the original Slammer worm. Further, the experiment set out to prove that the SSE routing worm was the fastest worm of its kind. Lastly, this research examined whether the computing systems architecture of today would facilitate a much more aggressive worm than has been observed in previous outbreaks.

This research found that despite an observable pattern of generation, IP addresses produced by Slammer are in a uniform distribution across the address space. The detailed examination of the IP address generation data set proved again, despite the presence of an observable generation pattern, that the second IP octet generated by the Slammer code for this experiment was also uniformly distributed across the address space. Finally, the third and fourth octets were shown to have no observable random number generation pattern, and they were uniformly distributed across their address space.

After establishing randomness, the Matlab model simulations were compared to and validated by the previously created infection rate models used by Zou for the Code Red, BGP routing worm, “/8” routing worm, the original Slammer worm, and the Slammer routing worm [ZTG05]. The infection doubling rate observed by Moore during the original Slammer worm release and the research of the original Slammer worm infection rate performed by Wei provided further validation of the Matlab model and Slammer worm infection rate [MPW03] [WMS05]. During the validation of the Matlab

model simulations, one of the experiment graphs provided by Zou was either an error or notated incorrectly. The Zou Slammer routing worm infection rate curve was significantly faster than what is expected for a worm operating with its characteristics. However, the analysis of the Slammer routing worm as hypothesized by Zou proved to be faster using the Matlab model simulations than the original Slammer worm.

The extension of this research to include the current speed of the computing systems shows worms would be significantly faster on a computing network of today than they were in 2003. The research showed that the worm infection rate for a worm in 2003 was increased by a factor of 3.6 times for a worm operating on a computing system of 2007 due to the increased packet generation rate. This research has given strong evidence that any scanning worm released on the architecture of today would cause even greater harm to the Internet infrastructure through its speed of infection and network congestion.

Finally, the new SSE routing worm is faster than any of the worms evaluated. The SSE routing worm was more than three times faster than the original Slammer worm and more than two times faster than the Slammer routing worm proposed by Zou. An SSE routing worm released today would have an infection rate 3.6 times faster than if it had been released in 2003 due to the faster infection packet generation.

## **5.2 Contributions and Significance of Research**

This research has furthered the understanding of the operation characteristics of scanning worms on an IPv4 network and laid the groundwork for future experiments into live worm research. The routing worms proposed by Zou have been validated and extended to an even faster version of routing worm illustrating that these worms pose a

great threat to the computing community and deserve further research. The expansion of this research into the speed of the current computing systems architecture exposes the fact that Slammer will most likely not remain the fastest worm and that there is a large void in the analysis of worms on current architectures. Through the use of the actual Slammer worm, a live host and a validated mathematical model, this experiment has furthered the research proposed by others.

### **5.3 Recommendations for Future Research**

There is a large void in the research of live worms on a network. Due to the problems incurred during the research of these worms several opportunities for future research based on this preliminary research are available. The largest area for continuation of research is to solve the issue of the auto-generated multicast address for UDP packets generated by Slammer, which afflicted every IP address generated. Also, the problem of modifying the Slammer assembly code to accept the changes required to set the IP address field with the “hit list” values needs to be investigated. These two problems may be related. Once these problems are solved many more research avenues open up.

The observation and testing of live worms on a network is an area that could validate many mathematical models currently in use by researchers worldwide. With the speed of computing systems continually increasing, these models need current data harvested from a live network to validate and ensure they incorporate the capabilities of today’s systems. As shown in this experiment, the current architecture is significantly faster than what was in place in 2003, which is the timeframe of when the majority of the worm research is based.

Finally, modifying the worms to operate on an IPv6 network and analyzing their ability to propagate in that environment would be groundbreaking research.

Incorporating the increase in speed of computing systems, the capabilities of the routing worms examined in this research and calculating the increase of vulnerable systems for a given software vulnerability would provide a significant leap forward in the research of self-propagating worms on an IPv6 network. While other researchers have made claims that the conversion to IPv6 would all but eliminate the capability of a scanning worm to propagate, the proof on an existing system with live worms and the characteristics of the computing systems of today has yet to be completed.

#### **5.4 Summary**

This research has expanded the knowledge of the operation of scanning worms on an IPv4 network and proved that the Slammer routing worm and SSE routing worm is faster than any previously observed worm. The groundwork laid by this research provides a solid foundation for future research into the area of live worms on a network.

## Appendix A

This appendix covers the hardware and software used during the research to complete the experiment.

### A.1 Experiment Hardware

The computers used to facilitate the experiment are Dell Latitude Laptops and their specifications are shown in the Table 22. The specifications for the switch connecting the laptops together are provided in Table 23.

Table 22. Experiment Computer Specifications

Victim Machine	Attacking Machine
Dell Latitude D620	Dell Latitude D600
2 GB 533 MHz DDR2 RAM	512 MB
Intel Core Duo T2400 1.83Ghz	Intel Pentium M 1600 Mhz
80 GB 5400 RPM HD	30 GB 4200 RPM HD
Broadcom NetXtreme BCM5752 Gigabit Ethernet	Broadcom 570x Gigabit Integrated Controller

Table 23. Port Switch Specifications

Linksys SD205 10/100 Switch (5-port)
10/100 Mbps
Category 5 Ethernet
5 x RJ45 ports

## A.2 Experiment Software

The software used in this experiment is detailed in the Table 24 including the operating system version numbers.

Table 24. Experiment Software Versions

Microsoft Windows 2000 5.00.2195	Victim Machine
Microsoft SQL Server 8.00194	Victim Machine
Wireshark Network Analyzer 00.99.3	Both Machines
Netcat 1.11	Attacking Machine
Matlab 7.3.0 (R2006b)	Attacking Machine
Frhed 1.1.0	Attacking Machine

## Appendix B

This appendix covers in detail the code and operation of Slammer. The information contained in this section is available upon request.

## Appendix C

This appendix describes the generation of the infection rate simulation by the Matlab model used in this research.

### **C.1 Matlab Model Infection Rate Simulation Code**

The code below was used to generate all of the worm infect curves for comparison to the available data and previously described mathematical models. As defined in the comments of the code “N” is set to be the total number of available address for the scale of the test. The number of vulnerable systems was denoted by “n.” The maximum number of iterations, which is converted to seconds for final analysis, is represented by “M.” The value for “M” in these experiments is arbitrary as this experiment considers the entire vulnerable system space and it is set to a number beyond the expected infect iteration found by preliminary testing. Further, the code is set to “break” out of the current trial when the number of potential victims reaches zero. Note that there is an equal chance of any number being generated by the pseudo random number generator (PRNG), thus the number of vulnerable systems is reduced by one without regard to which number in the vulnerable range was guessed by the PRNG. “K” represents the number of trials. The number of trials is an arbitrary value set high enough to allow for the complete infection of all vulnerable systems. The Matlab code was set to exit the current trial after the last vulnerable system was infected to decrease the time between trails. The initial number of infected systems is represented by “I” and is highlighted in the code provided below. Table 39 shows the values used for the variations of the Matlab simulations.



Table 25. Matlab Model Experiment Variable Values

	N - # of IP addresses	n - # of Vulnerable Systems	M - # of Iterations	K - # of Trials	I - Initial # of Infected Systems
Code Red Worm	4,294,967,296	360,000	10,000,000	1	Ten
Zou Code Red "/8" Routing Worm	1,946,156,941	360,000	500,000	1	Ten
Zou Code Red BGP Worm	1,228,360,647	360,000	500,000	1	Ten
Zou Slammer Worm	4,294,967,296	100,000	10,000,000	20	Ten
Zou Slammer Routing Worm	1,946,156,941	100,000	500,000	20	Ten
Slammer Worm 2003	4,294,967,296	74,856	10,000,000	20	One
Slammer Worm 2007	4,294,967,296	74,856	10,000,000	20	One
Slammer Routing Worm 2003	1,946,156,941	74,856	500,000	20	One
Slammer Routing Worm 2007	1,946,156,941	74,856	500,000	20	One
SSE Routing Worm 2003	16,777,216	645	500,000	50	One
SSE Routing Worm 2007	16,777,216	645	500,000	50	One

```

*****

% N is the total # of IP addresses
% n is the total # of potential victims
% M is the maximum # of iterations
% K is the number of trials
% I (highlighted) is the initial number of infected systems – this value needs to manually changed in the
%      code
function [TargetInfectTime,VictimInfectTime] = IPsim(N,n,M,K) %Designates the function for Matlab
TargetInfectTime=M*ones(1,K); %Creates M arrays of ones the size of K for storage of TargetInfectTime
infected=zeros(1,K); %Creates an array of zeros the size of K for storage of infected
VictimInfectTime=zeros(K,n); % Creates an array of zeros K by n for storage of VictimInfectTime
for j=1:K
    n1=n; % Sets the upper limit of the vulnerable systems range equal to total number of potential victims
    for i=1:M
        IP = ceil(N*rand(1,(infected(j)+I))); % generates a random number in the range of N for each
                                                %infected system and turns it into an integer
        if sum(IP<=(n1+1))>=1 %if IP address is less than or equal to the upper value of the
                                % vulnerable systems enter loop
            VictimInfectTime(j,infected(j)+1:infected(j)+sum(IP<=(n1+1)))=
            i*ones(1,sum(IP<=(n1+1))); %Sets the time of infection for each IP address
                                    %within the vulnerable system range, this check is
                                    %performed multiple times if more than one is hit
            infected(j)=infected(j)+sum(IP<=(n1+1)); %Increases the number of infected systems
            n1=n1-sum(IP<=(n1+1)); %Reduces the number of vulnerable systems by number hit
        end
        if n1 == 0 % When number of Vulnerable systems reaches zero break
            break
        end
    end
end
results=fopen('results.txt','w'); % Opens “results.txt” for writing of data
fprintf(results,'Number of Machines Infected %10.0f\n',infected); % Prints # Infected to file
fprintf(results,'Victim Infected %10.0f\n', VictimInfectTime); % Prints Infect Time to file
fclose(results) % Closes “results.txt”
end
end
*****

```

## Appendix D

This appendix contains a detailed breakout of a typical UDP header from a Slammer packet. This information is available upon request.

## Appendix E

This appendix contains the detailed information for the creation of the SSE routing worm as previously discussed in Section 4.5. The information in this appendix is available upon request.

## Bibliography

- [Bla06] Black, Paul E. "Pseudo-Random Number Generator," in Dictionary of Algorithms and Data Structures. [online]. 16 October 2006.  
<http://www.nist.gov/dads/HTML/pseudorandomNumberGen.html>.
- [CER02] CERT/CC. "CERT Advisory CA-2001-13 Buffer Overflow In IIS Indexing Service DLL." [online]. 17 January 2002.  
<http://www.cert.org/advisories/CA-2001-13.html>
- [CGK03] Chen, Zesheng, Lixin Gao, Kevin Kwiat. "Modeling the Spread of Active Worms" in 2003 IEEE Conference on Open Architectures and Network Programming. 2003. Pages 1890-1900.
- [EEy03] eEye Digital Security. "Sapphire Worm Code Disassembled." [online]. 27 January 2003. <http://www.eeye.com/html/Research/Flash/sapphire.txt>
- [Fes04] Festa, Paul. "Microsoft: To secure IE, upgrade to XP." [online]. 23 September 2004.  
[http://news.com.com/Microsoft+To+secure+IE%2C+upgrade+to+XP/2100-1032\\_3-5378366.html](http://news.com.com/Microsoft+To+secure+IE%2C+upgrade+to+XP/2100-1032_3-5378366.html)
- [FIP01] Floyd, Sally, Vern Paxson. "Difficulties in Simulating the Internet" in IEEE/ACM Transaction on Networking. August 2001. Vol. 9, No. 4, Pages 392-403.
- [Gro02] Gromov, Gregory R. "History of Internet and WWW: The Roads and Crossroads of Internet History." [online]. 2002.  
<http://www.netvalley.com/intvalstat.htm>.
- [Haa99] Haahr, Mads. "Introduction to Randomness and Random Numbers." [online]. June 1999. <http://www.random.org/essay.html>.
- [Han03] Hansen, Joshua C., "Worm Propagation in Heterogeneous Networks: Weaknesses in the National Strategy for Cyberspace Protection." Thesis Naval Postgraduate School, Monterey CA. March 2003.
- [Hal06] Halfhill, Tom R. "The Mythology of Moore's Law." [online]. September 2006.  
[http://www.ieee.org/portal/site/sscs/menuitem.f07ee9e3b2a01d06bb9305765bac26c8/index.jsp?&pName=sscs\\_level1\\_article&TheCat=2165&path=sscs/06Sept&file=Halfhill.xml](http://www.ieee.org/portal/site/sscs/menuitem.f07ee9e3b2a01d06bb9305765bac26c8/index.jsp?&pName=sscs_level1_article&TheCat=2165&path=sscs/06Sept&file=Halfhill.xml)

- [Hei04] Heidari, Mohammad. "Malicious Codes in Depth." [online]. 29 November 2004. <http://www.securitydocs.com/library/2742>
- [Hof90] Hoffman, Lance J., editor, "Rogue Programs: Viruses, Worms, and Trojan Horses", Van Nostrand Reinhold, New York, New York. 1990. Page 7.
- [Huf06] Huffaker, Bradley. "IPv4 BGP Geopolitical Analysis." [online]. 2 March 2006. <http://www.caida.org/analysis/geopolitical/bgp2country/index.xml>.
- [HyE03] Hypponen, Mikko, Erdelyi Gergely. "F-Secure Virus Descriptions: Slammer." [online]. 25 January 2003. <http://www.f-secure.com/v-descs/mssqlm.shtml>
- [IWS07] Internet World Stats. "Internet Growth Statistics." [online]. 10 January 2007. <http://www.internetworldstats.com/emarketing.htm>.
- [KeW91] Kephart, Jeffery O., Steve R. White. "Directed-Graph Epidemiological Models of Computer Viruses" in Proceedings of the 1991 IEEE Computer Society Symposium on Research in Security and Privacy. May 1991. Pages 343-359.
- [KiE03] Kienzle, Darrell M., Matthew C. Elder. "Recent Worms: A Survey and Trends" in Proceedings of the ACM CCS Workshop on Rapid Malcode (WORM'03). 27 Oct 2003. Pages 1-10.
- [KRD04] Kim, Jonghyun, Sridhar Radhakrishnan, Sudarshan K. Dhall. "Measurement and Analysis of Worm Propagation on Internet Network Topology" in Consumer Communications and Networking Conference (CCNC). 3 January 2004. Pages 495-500.
- [Lem03] Lemos, Robert. "Counting the Cost of Slammer." [online]. 31 January 2003. [http://news.com/Counting+the+cost+of+Slammer/2100-1001\\_3-282955.html](http://news.com/Counting+the+cost+of+Slammer/2100-1001_3-282955.html).
- [Lit02] Litchfield, David. "Threat Profiling Microsoft SQL Server (A Guide to Security Auditing)." [online]. 20 July 2002. [www.ngssoftware.com](http://www.ngssoftware.com).
- [Mar04] Martin, Jeremy. "Information Systems Security Training Virus and Worms." [online]. 5 Oct 2004. <http://www.securitydocs.com/library/2627>

- [MSB02] Moore, David, Colleen Shannon, Jeffery Brown. "Code-Red: a case study on the spread and victims of an Internet worm." [online]. 2002. <http://www.caida.org/publications/papers/2002/codered/codered.pdf>
- [MPS03] Moore, D., Paxson, V., Savage, S., Shannon, C., Staniford, S., and Weaver, N. 2003. "Inside the Slammer Worm." *IEEE Security and Privacy*. July/August 2003. Pages 33-39.
- [MPW03] Moore, D., Paxson, V., Savage, S., Shannon, C., Staniford, S., and Weaver, N. "The Spread of the Sapphire/Slammer Worm." [online]. 6 November 2006. <http://www.caida.org/publications/papers/2003/sapphire/sapphire.html>.
- [Mur03] Murphy, Matthew. "Analysis of Sapphire SQL Server Worm." [online]. 26 January 2003. <http://student.missouristate.edu/m/matthew007/research/virus/sqlworm.asp>.
- [Odl03] Odlyzko, Andrew M. "Internet traffic growth: Sources and implications" in Optical Transmission Systems and Equipment for WEM Networking II. 2003. <http://www.dtc.umn.edu/~odlyzko/doc/itcom.internet.growth.pdf>. Pages 1-15.
- [PeS04] Perumalla, Kaylan S., Srikanth Sundaragopalan. 2004. "High-Fidelity Modeling of Computer Network Worms" in IEEE Proceedings of the 20<sup>th</sup> Annual Computer Security Applications Conference (ACSAC'04). 6 December 2004. Pages 126-135.
- [RSL04] George F. Riley, Monirul I. Sharif, Wenke Lee. "Simulating Internet Worms" in IEEE Proceedings of The IEEE Computer Society's 12th Annual International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunications Systems (MASCOTS'04). 1 October 2004. Pages 268-274.
- [ShM04] Shannon, Colleen, David Moore. "The Spread of the Witty Worm" in *IEEE Security & Privacy*. July/August 2004. Pages 46-50.
- [Sla95] Slade, Robert, "Robert Slade's Guide to Computer Viruses", Springer-Verlag, New York, New York, 2nd Ed., 1995. Page 52.
- [Sto03] Stone, C. "Worm-Annotated." [online]. January 2003. <http://www.boredom.org/~cstone/worm-annotated.txt>.

- [WaC02] Waddington, Daniel G., and Fangzhe Chang. "Realizing the Transition to IPv6." *IEEE Communications Magazine*. June 2002. Pages 138-148.
- [WDP03] Wagner, Arno, Thomas Dubendorfer, Bernhard Plattner, and Roman Hiestand. "Experiences with Worm Propagation Simulations" in *Proceedings of the ACM CCS Workshop on Rapid Malcode (WORM'03)*. 27 OCT 2003. Pages 34-41.
- [Whe02] Whelan, Micheal. "Mining Internet Traffic Records for Geographic Information." [online]. September 2002. <http://www.e-insights.com/general/Geostudy.doc>.
- [WMS05] Songjie Wei, Jelena Mirkovic, Martin Swany. "Distributed Worm Simulation with a Realistic Internet Model" in *IEEE Proceedings of the Workshop on Principles of Advanced and Distributed Simulation (PADS'05)*. 1-3 June 2005. Pages 71-79.
- [YuW04] Yu, Wei. "Analyzing the Performance of Internet Worm Attack Approaches" in *Consumer Communications and Networking Conference (CCNC)*. 3 January 2004. Pages 501-506.
- [ZTG02] Zou, Cliff Changchun, Weibo Gong, Don Towsley. "Code Red Worm Propagation Modeling and Analysis." (CCS'02) 18-22 November 2002. Pages 138-147.
- [ZTG05] Zou, Cliff C., Don Towsley, Weibo Gong, Songlin Cai. "Routing Worm: A Fast, Selective Attack Worm based on IP Address Information" in *IEEE Proceedings of the Workshop on Principles of Advanced and Distributed Simulation (PADS'05)*. 1-3 June 2005. Pages 199-206.



## Vita

James Gorsuch was born in Berea, Ohio, in 1969 and graduated from North Ridgeville High School in 1987. He enlisted in the United States Air Force in 1987 and served 15 years earning the rank of Technical Sergeant. During his enlisted service, he worked as a Defensive Avionics, Communications, and Navigations System technician on the B-1 and B-52 bombers before being selected for the B-2 program at Edwards AFB. While working on the B-2, he became a fully qualified crew chief in addition to his normal avionics systems responsibilities. Lt Gorsuch, then Staff Sergeant, moved to Eglin AFB to work with the 53<sup>d</sup> Wing to test and develop the defensive avionics software for the B-2.

In 2003, Lt Gorsuch was selected to attend Officer Training School at Maxwell AFB where he earned his commission. Lt Gorsuch's first assignment as an officer was to lead the Intrusion Detection Team at Headquarters Air Force Materiel Command at Wright-Patterson AFB where he also served as the Executive Officer to the A6. Lt Gorsuch was selected to attend the Air Force Institute of Technology (AFIT) in August 2003. Lt Gorsuch will be assigned to Maxwell AFB, Alabama providing the Air Force Wargaming Institute (AFWI) with leadership and focus for all joint campaign-level combat simulation software.

REPORT DOCUMENTATION PAGE				Form Approved OMB No. 074-0188	
<p>The public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of the collection of information, including suggestions for reducing this burden to Department of Defense, Washington Headquarters Services, Directorate for Information Operations and Reports (0704-0188), 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to a penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number.</p> <p><b>PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.</b></p>					
1. REPORT DATE (DD-MM-YYYY) 22-03-2007		2. REPORT TYPE Master's Thesis		3. DATES COVERED (From – To) March 2006 - March 2007	
4. TITLE AND SUBTITLE ANALYSIS OF ROUTING WORM INFECTION RATES ON AN IPV4 NETWORK				5a. CONTRACT NUMBER	
				5b. GRANT NUMBER	
				5c. PROGRAM ELEMENT NUMBER	
6. AUTHOR(S)  Gorsuch James, First Lieutenant, USAF				5d. PROJECT NUMBER	
				5e. TASK NUMBER	
				5f. WORK UNIT NUMBER	
7. PERFORMING ORGANIZATION NAMES(S) AND ADDRESS(S)  Air Force Institute of Technology Graduate School of Engineering and Management (AFIT/EN) 2950 Hobson Way WPAFB OH 45433-7765 DSN: 785-3636				8. PERFORMING ORGANIZATION REPORT NUMBER  AFIT/GCS/ENG/07-04	
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)				10. SPONSOR/MONITOR'S ACRONYM(S)	
				11. SPONSOR/MONITOR'S REPORT NUMBER(S)	
12. DISTRIBUTION/AVAILABILITY STATEMENT  APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED					
13. SUPPLEMENTARY NOTES					
14. ABSTRACT <p>Malicious logic, specifically worms, has caused monetary expenditure problems to network users in the past. Worms, like Slammer and Code Red, have infected thousands of systems and brought the Internet to a standstill. This research examines the ability of the original Slammer worm, the Slammer based routing worm proposed by Zou et al, and a new Single Slash Eight (SSE) routing worm proposed by this research to infect vulnerable systems within a given address space. This research investigates the Slammer worm's ability to generate a uniform random IP addresses in a given address space. Finally, a comparison of the speed increase from computing systems available today versus those in use during the original Slammer release is performed.</p> <p>This research finds that the both the Slammer based routing worm and the SSE routing worm are faster than the original Slammer. The random number generator of the original Slammer worm does generate a statistically uniform distribution of addresses within the range under test. Further, this research shows that despite the previous research into the speed of worm propagation, there is a large void in testing worms on the systems available today that need to be investigated. The speed of the computing systems that the worms operated on in the past were more than three times slower than today's systems. As the speed of computer systems continue to grow, the speed of worm propagation should increase with it as their scan rates directly relate to their infection rate. As such, the immunity of the future IPv6 network, from scanning worms may need to be reexamined.</p>					
15. SUBJECT TERMS Malicious Logic, Worm, SQL Slammer, Sapphire, Slammer, Routing Worm, Propagation, Infection Rate, Pseudo Random Number Generator, Border Gateway Protocol, Vulnerability, User Datagram Protocol, Classless Inter-Domain Routing, Epidemiological, Internet Protocol, IPv4,					
16. SECURITY CLASSIFICATION OF:		17. LIMITATION OF ABSTRACT		18. NUMBER OF PAGES	
REPORT U	ABSTRACT U	c. THIS PAGE U	UU	129	19a. NAME OF RESPONSIBLE PERSON Barry E. Mullins, Ph.D. (ENG)
				19b. TELEPHONE NUMBER (Include area code) (937) 255-3636 ext. 7979; email: Barry.Mullins@afit.edu	

Standard Form 298 (Rev: 8-98)  
Prescribed by ANSI Std. Z39-18